МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Катерина
НЕСТЕРЕНКО
"____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: "Програмна система аналізу вмісту аудіофайлів"

Виконавець: студент групи ПІ-222М Хілько Микола Миколайович

Керівник: к.т.н. Семко Олексій Вікторович

Нормоконтролер: ст.в Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2023 р

ЗАВДАННЯ
на виконання дипломного проекту студента
Хілько Миколи Миколайовича

1. Тема проекту: «Програмна система аналізу вмісту аудіофайлів»
затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2023 р. по 31.12.2023 р.

3. Вихідні данні до проекту: програмний продукт розробити у вигляді веб-додатку для браузеру.

4. Зміст пояснювальної записки:
   1. Аналіз проблеми дослідження вмісту аудіофайлів та порівняння винайдених рішень
   2. Характеристика основних підходів до аналізу вмісту аудіофайлів
   3. Структура та реалізація програмної системи
   4. Тестування та випробування програмної системи

5. Перелік обов'язкових слайдів презентації:
   1. Актуальність розробки програмної системи
   2. Вимоги до системи
   3. Розпізнавання мовлення
   4. Пошук ключового слова
   5. Дослідження контексту
   6. Важливість графічного інтерфейсу
   7. Графічний інтерфейс програми
   8. Робота програми
   9. Напрямки удосконалення

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку роботи дипломного проектування Написання 1 розділу, представлення керівнику	02.10.23 – 08.10.23	
2.	Попередній друк 1 розділу та допоміжних сторінок (чорновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль.	09.10.23 – 22.10.23	
3.	Написання 2 розділу, представлення керівнику	23.10.23 – 05.11.23	
4.	Написання 3 розділу, представлення керівнику	06.11.23 – 19.11.23	
5.	Написання 4 розділу, представлення керівнику	20.11.23 – 03.12.23	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	04.12.23 – 10.12.23	
7.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки.	11.12.23 – 16.12.23	
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	17.12.23 – 18.12.23	
9.	Отримання відгуку керівника, рецензії.	18.12.23 – 19.12.23	
10.	Передзахист роботи (наявність відрукованої ПЗ, презентації, позитивного відгуку керівника)	20.12.23	
11.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	21.12.23 – 22.12.23	
12.	Захист дипломної роботи перед ЕК	27.12.23	

7. Дата видачі завдання 02.10.23
Керівник: к.т.н. Семко Олексій Вікторович
Завдання прийняв до виконання: Микола ХІЛЬКО
Дата: 12.12.22

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмна система аналізу вмісту аудіофайлів»: 83 с., 29 рис., 7 табл., 25 інформаційних джерел.

АУДІОФАЙЛ, РОЗПІЗНАВАННЯ МОВЛЕННЯ, ПОШУК КЛЮЧОВИХ СЛІВ, ДОСЛІДЖЕННЯ КОНТЕКСТУ

Об’єкт розробки – процеси аналізу вмісту аудіофайлів.

Мета роботи – спрощення, прискорення та автоматизація аналізу вмісту аудіофайлів з можливістю контекстуального пошуку.

ABSTRACT

Explanatory note to the thesis " Software system for analysis of the contents of audio files ": 83 p., 29 Fig., 7 table., 25 information sources.

AUDIO FILE, SPEECH RECOGNIZING, SEARCH KEYWORDS, CONTEXT RESEARCH

Property development – audio file content analysis processes.

Purpose – Simplification, acceleration, and automation of audio file content analysis with the possibility of contextual search.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

HDD (Hard disk drive) – жорсткий диск або вінчестер, запам’ятовувальний пристрій.

SSD (Solid-state drive) – запам’ятовувальний пристрій на основі мікросхем, має контролер для керування мікросхемами пам’яті.

M.2 – запам’ятовувальний пристрій який має формат PCI-E, найшвидший тип пам’яті для електронно обчислювальних машин на сьогоднішній день.

GUI (Graphic user interface) – графічний інтерфейс користувача.

UX (User experience) – методика побудови дизайну на основі думки користувача.

UML (Unified Modelling Language) – уніфікована мова моделювання для парадигми об’єктно-орієнтованого програмування.

(К, М, Г) Б – кількість байт, одиниця виміру пам’яті, має в собі 8 бітів, найменшу одиницю пам’яті, приставка Кіло як 1000 (1 тисяча) байт, Мега як 1000000 (1 мільйон) байт, Гіга як 1000000000 (1 мільярд) байт.

ГГц – гіга герц, герц, одиниця виміру частоти оперативної одиниці, Гіга як 1000000 герц.

ШІ – штучний інтелект.

МН – машинне навчання.

ВСТУП

Система аналізу вмісту аудіофайлів спрямована на оптимізацію та спрощення процесу обробки звукової інформації, щоб полегшити роботу людини при аналізі зв’язного або незв’язного мовлення, що міститься у звукових записах. Основний упор робиться на визначення ключових фрагментів або слів у звукових доріжках, а також пошуку визначень чи уточнень до них. Цей процес аналізу аудіофайлів спрямований на підтримку та полегшення розуміння змісту звукової інформації.

Основна функція системи полягає в ефективному визначенні і розрізненні частин аудіофайлів, що мають ключове значення або важливий зміст. Ця система сприяє зручнішому аналізу аудіоданих, полегшуючи виявлення конкретних слів, фраз чи важливих виразів у звукових записах.

Такий інструментарій є важливим у сферах, де необхідно ретельно аналізувати вміст аудіофайлів, таких як мовні дослідження, звукова аналітика чи інші сценарії, де зміст звукових даних має велике значення, а саме: органам правозахисту – пошук та представлення доказів з аудіозаписів, охоронним фірмам – перегляд та аналіз з приборів спостереження, центрах обслуговування клієнтів та обробки дзвінків – швидка обробка інформації клієнтів та сповіщення про можливі порушення.

Сучасність потребує пришвидшення роботи у всьому інформаційному середовищі, звукової та візуальної інформації стає більше ніж текстової, її аналіз має розвиватись з більшою швидкістю аніж вона надходить для того щоб встигати обробляти весь об’єм. Порівняємо наповнення та обробку текстової інформації та звукової. На наповнення текстової інформації, а саме написання чи друкування тексту, потрібно більше часу ніж на обробку, тобто читання, за рідкими виключеннями. А для наповнення звукової інформації, запис мовлення, потрібно стільки ж часу скільки й на прослуховування. Це свідчить про те що обробка звукової інформації виділеної однією людиною, може бути оброблена за той самий час іншою людиною, в той час коли існуючі технології комп’ютерної та програмної інженерії дозволяють виконувати багато процесів одночасно з математично вирахуваною точністю.

З цього випливає мета та метод дослідження – спрощення, прискорення та

автоматизація аналізу вмісту аудіофайлів за допомогою розробки зручного додатку для визначення співпадінь ключових слів та пошуку контексту для розуміння змісту звукової інформації.

У результатах роботи велика увага приділялась зручності та ефективності користування за допомогою найкращих практик користувацького досвіду (user experience). До того ж були проведені відповідні дослідження за методикою функціональних точок та наведена таблиця успішності практик. Проведені дослідження актуальності та ефективності використаних алгоритмів.

Розробка та дослідження проводилися під управлінням операційної системи Windows. Розробка проводилася на мовах програмування Kotlin та React Typescript з використання бібліотеки Material UI.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ ВМІСТУ АУДІОФАЙЛІВ ТА ПОРІВНЯННЯ ВИНАЙДЕНИХ РІШЕНЬ

1.1 Поняття аналізу вмісту аудіофайлів

Аналіз вмісту аудіо файлів є складною процедурою, що не лише ідентифікує основні елементи, але й розкриває їхні зв’язки з іншими граматичними структурами, а також визначає контекст і характеристики використання звукових компонентів у різних сценаріях, таких як вимова чи гучність.

Слід зазначити, що аудіофайли є важливими носіями даних у сучасних інформаційних системах. Вони містять у собі значну кількість інформації, представлену у вигляді звукових сигналів, які можна інтерпретувати та розуміти як для комп'ютерів, так і для людей. Основне відмежування аудіо файлів від текстових полягає у їхній природі: текстові файли зберігають дані у вигляді символів і кодувань, що дозволяють легкої конвертації у звичайний текст, тоді як бінарні файли організовані іншим чином, використовуючи інші методи подання інформації, що не прив'язані до символьних наборів. Саме складне інтерпретування робить аналіз звукової інформації більш складним.

Звук для обчислювальних систем – це аналогові звукові хвилі, які перетворюються на цифровий формат для зберігання та обробки. Звук складається з коливань атмосферного тиску, які потім збираються та конвертуються у цифрові дані, які можуть бути зрозумілі та оброблені системою за відповідними форматами.

Операційні системи зазвичай працюють з цифровим звуком, який представлений у вигляді цифрових значень, або бітів. Звукові файли, такі як WAV, MP3 або FLAC, зберігають звукові дані у вигляді числових значень, що представляють амплітуду звукових хвиль у різний час.

Для того, щоб система могла відтворити звук, ці цифрові значення перетворюються на аналогові звукові сигнали. Зазвичай це відбувається через аудіо пристрої, такі як звукові карти, які отримують цифрові дані та перетворюють їх у

звукові хвилі, які можна почути через динаміки або навушники. Цей процес конвертації включає аналого-цифрове перетворення (ADC) при записі аудіо та цифро-аналогове перетворення (DAC) під час відтворення, де цифрові дані перетворюються на аналогові звукові сигнали для відтворення звуку.

Зрозумівши, що собою представляє аудіофайл та як зберігається звук в електронно-обчислювальних системах можна детальніше розібрати етапи аналізу.

Розпізнавання мовлення. Фундаментальна частина аналізу – це отримати предмет аналізу. Проблема полягає в тому що аудіофайл зберігає в собі біти які представляють відформатовану звукову хвилю, яка дуже відрізняється від інших, а слово для пошуку представляє собою біти які форматуються в загальновизначену систему за допомогою кодувань. Потрібно звести все до спільного дільника, і тут постає основна проблема, як перетворити формат відображення даних, який при різних обставинах може становити різні відображення для одного значення в статичний, який однаковий і незалежний від перетворення, або навпаки. Вирішення цієї проблеми є ключовим для даної системи.

Пошук співпадінь. Вирішення проблеми розпізнавання приведе до нової проблеми, а саме як визначити суб’єкти аналізу, те від чого аналіз буде відштовхуватись. Ця частина аналізу дає змогу зменшити об’єм неавтоматизованої роботи та показати точне місце очікуваного елемента. Ускладнення полягає в минулій проблемі але вже збоку пошуку, оскільки слова можуть відмінятися та залежать від роду та часу використання. Отже, доцільніше всього називати саме співпадіннями, та оцінювати якість цього схожості шуканого слова чи словосполучення з тим що є наявним у предметі аналізу. Цей етап є одним з найпрактичніших серед інших, оскільки він робить якраз ту роботу яку зазвичай роблять люди та має надзвичайний потенціал у автоматизації, як у швидкості, так і у якості.

Дослідження контексту. Дослідження контексту виявляється критичним у аналізі мовлення, як зв’язного, так і незв’язного. Після пошуку співпадінь, перехід до обробки змісту контексту стає наступним кроком аналізу. Ці слова, розкидані у тексті, можуть набувати різних відтінків залежно від контексту, як правило слів які

найближче знаходяться до ключового. Не лише слово важливе, але й те, що оточує його. Близькі за значенням слова чи фрази можуть розкрити більше про об'єкт аналізу. Це дає змогу знаходити частки вираженого змісту та виокремлювати об'єкти для подальшого більш детального аналізу. Ця інформація буде потрібна для людини для визначення доцільності співпадіння для подальшої обробки. Отже, ця фаза дослідження виявляється такою ж важливою, як і попередня.

Визначення характеру та мотиву. Розкриття характеру - це ще одне місце, де аналіз стає складним і вимагає пильності. Виражена зв’язними між собою словами мова, може відображати різні характеристики та почуття, як позитивні, так і негативні. Хоча аналіз цього аспекту потребує додаткового дослідження в контексті та часто не такий ефективний через можливість надмірних зусиль, але в певних сферах він має перевагу у дослідженнях. Ця фаза аналізу ускладнюється надзвичайною різноманітністю емоцій та їх вираження різними типами людей. Цей етап надзвичайно залежить від людського фактору, тому дослідження зводиться до того чи взагалі можливе вирішення такого завдання.

Представлені етапи аналізу вмісту аудіофайлів і є загальним предметом дослідження та ціллю створення програмної системи.

1.2 Аналіз стану проблеми дослідження аудіофайлів

Проблема дослідження аудіофайлів полягає в тому що обробка вмісту такого файлу займає більше часу аніж його створення, на відміну від текстових чи візуальних даних. Це випливає з логічних вимірювань часу: письмо займає більше часу ніж читання, так само як і малювання чи графічне зображення вимагає більше часу у порівнянні з візуальною оцінкою, а ось запис звуку займає однаково часу так само як і прослуховування, а з аналітичними діями щодо аудіозапису цей час ще збільшується в рази. Варто також враховувати якість запису, гучність, вимову, вади звуку і так далі. Коли з іншими форматами інформації в електронно-обчислювальних системах таких проблем не виникає, вони набагато більш розвинені за формат аудіо. Текст обмежений кодуваннями, відсутність людського

фактору такого як почерк та розмір тексту, зображення можуть відрізнятись лише в залежності від якості, проте зараз технології по покращенню якості зображень також на висоті. Але обробка аудіо залишається найменш розвиненою та складною, а розвиток використання таких записів збільшується з кожним роком. До таких форматів також можна віднести і відео, оскільки відеофайл має звукову доріжку, і за потреби її можна виокремити та конвертувати відеофайл в аудіофайл.

Відео та аудіо інформація поширюється дуже швидко через їх унікальну здатність передавати здебільшого повністю вірогідні дані з вмістом зображення та звуку, який фіксує все що відбувається навколо. За допомогою них люди можуть спілкуватися незалежно від відстані та мати всі переваги особистої зустрічі. Соціальні мережі наповнені звуковим контентом в усьому, починаючи з персональних блогів, закінчуючи навчанням чи науковими дослідами. Новини супроводжуються звуковою та візуальною інформацією задля представлення вірогідних та миттєвих даних про ситуацію. Будь-які дії переважно потребують мати підтвердження у відеоформаті.

Таким чином збільшується кількість носіїв з аудіо та відео вмістом а заразом з цим збільшується потреба в аналізі цих даних. Докази різних злочинів зазвичай зберігається саме в відео та аудіофайлах. Камери відеонагляду, запис звуку переговорів, ті ж самі повідомлення в соціальних мережах, можуть приховувати в собі інформацію яка може бути корисна для фіксації порушень. Але як було досліджено цю інформацію людськими силами оброблювати занадто неефективно.

Автоматизація таких процесів зможе значно зменшити людей з цих процесів, а невдовзі можливо взагалі виключити, переведення аналізу на вираховувальні потужності електронно-обчислювальних систем призведе до своєчасно та детального аналізу. Проте зараз виключити людський фактор обробки неможливо, оскільки методи обробки мають певну точність яка лише зрідка дорівнює 100 відсоткам, тому варто визначити складність застосування методів обробки аудіофайлів.

Методи обробки звуку є надзвичайно складними у використанні. Оскільки звук є хвилею то до його аналізу відносяться багато математично складних аналізів,

наприклад спектральний аналіз, хвильовий формат, спектрограми та акустичний аналіз. Всі ці методи можуть бути використані для будь яких хвиль, проте ними займаються в лабораторіях спеціалісти з вищою освітою та багатьма роками досвіду. Складність таких процесів максимальна та вони використовуються переважно в наукових дослідах чи занадто складних випадках аналізу звуку.

Та є інший спосіб, це використання штучного інтелекту або нейронних мереж через спеціально підготовані моделі. Такий метод має переважно набагато меншу точність, проте меншу складність та рівень підготовки спеціалістів до використання. Проте тепер цю задачу мають виконувати програмні інженери та ставати вчителями для нейромереж. Таке навчання також займає немало часу, проте ці знання нікуди не зникнуть з часом та будуть тільки розвиватись з кількість нових правильно та неправильно оброблених даних.

1.3 Аналітичне дослідження систем аналізу вмісту аудіо файлів

Як було зазначено вище актуальність проблеми є дуже висока, і схожі за ціллю програмні системи є в наявності в глобальній мережі. Більшість таких програм мають аналіз аудіофайлів як додатковий функціонал, проте він там представлений в дуже стислому форматі. А також є ряд програм які позиціонують як повний аналізатор звуку в файлах, вони проводять надзвичайно математично складні аналізи звуку і як правило не використовуються для пошуку співпадінь слів у мовленні. До того ж більшість програм є платними чи з оплачуваними підписками для такого функціоналу, їх розглянуто не буде. Отож під пошук наявних систем підпадають безкоштовні програми з наявним функціоналом пошуку співпадінь слів у аудіофайлах чи перетворенні змісту аудіофайлу в текст та можливий пошук по ньому. Програми відібрані з різними алгоритмами розпізнавання мовлення, алгоритмами пошуку, розширеннями типів файлів, розмірами додатку, підтримуваними операційними системами та платформами. До того ж важливою оцінкою будуть залишатись засоби представлення інтерфейсу та його зручність.

Для тестування обрано аудіофайл розміром 10Мб та довжиною запису в 4

хвилини та 30 секунд. Проте для виконання порівнянь можуть застосовуватись і інші файли з різною довжиною запису, оскільки порівняння буде відбуватися перевіркою співвідношення часу обробки до часу запису.

Braina – Human Language Interface

В описі програми вказаний такий функціонал:

- Перетворення аудіо файлів в текстовий формат та навпаки
- Мовні налаштування
- Можливість програвання аудіо та відеофайлів
- Пошук інформації з інтернету
- Помітки
- Асистент у вигляді штучного інтелекту

Це десктопна програма та підтримує операційну систему Windows, IOS та Android. Для мобільних платформ доступний у відповідних магазинах безкоштовна. Має розширений функціонал за платною підпискою. Інсталятор має вагу 320Мб, а сама програма 340Мб. При цьому якщо користування програмою передбачає аналіз аудіофайлів, то потрібно інсталювати мовний пакет вагою в 2Гб.

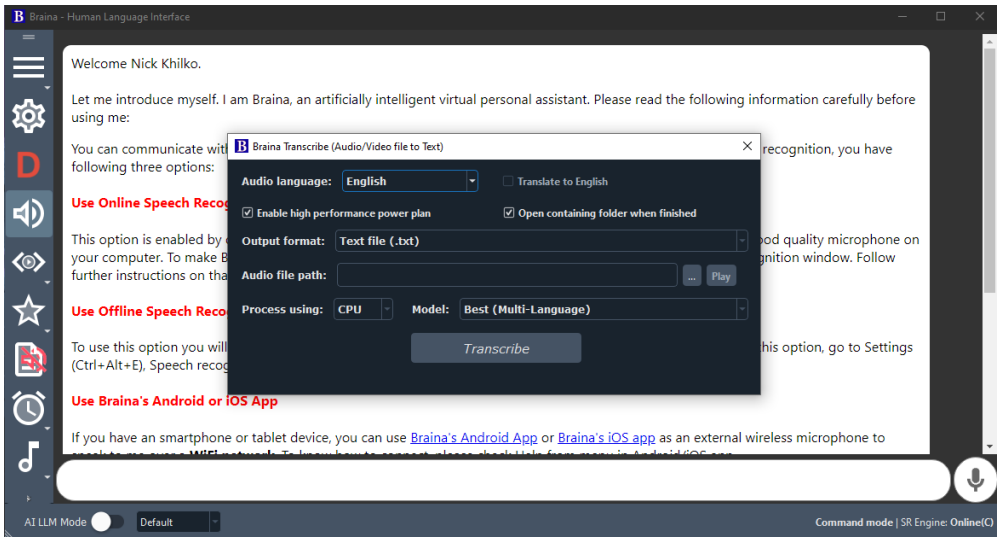

Рис.1.3.1 інтерфейс програми Braina

Після запуску програми відкривається вікно з привітанням та описом функціоналу, зліва наявне меню використання функціоналу. Більшість функціоналу є дуже дивним: пошук в інтернеті, годинник, нагадування та інше. Основний функціонал вміщено в одну вкладку зі звуком. Програма привітала голосовим

сповіщення штучного інтелекту та провела інструкцію.

Перейдемо до функціонального тесту, тест пройшов за час відповідний часу відтворення файлу, тобто 4 хвилини 30 секунд. Варто визначити що через встановлення мовного пакету можна зробити висновок що ця програма використовує мовні чи так звані нейромережні моделі, за якими перетворює звук в текст. І це максимальна швидкість аналізу, оскільки штучний інтелект ніби відтворює файл без прискорення для розпізнавання мовлення. Та звісно якість перетворення вражаюча, мовна модель на 2Гб справляється на 100 відсотків з поставленою задачею. Всі слова були відтворені правильно, але немає жодного пошуку по слова, навіть текст представлений немає функції пошуку, його потрібно скопіювати і аналізувати в іншій програмі.

Інтерфейс є дуже не зручним на головній панелі, іконки в меню незрозумілі та мають підказки з описаною функціональністю, хоча текст міг би вміститись поруч з іконками. Весь інший функціонал представлений в випливаючих діалогах, що не дозволяє сховати інтерфейс програми чи виконувати будь які дії поза діалоговим вікном, хоч в програмі, хоч в операційній системі. Зручним відкриттям став словник, в який можна додавати свої слова та переглянути наявні в мовній моделі, проте мовну модель додавання слова не розширить та не навчить, тому не дуже є сенс такої можливості.

Otter

Програма представляє собою календар з нотатками повністю керований за допомогою голосу та з переважним збереженням саме аудіозаписів. Вони безкоштовна проте доступна лише для мобільних девайсів IOS чи Android. Програма має надзвичайну кількість гарних відгуків про найкращу роботу з аудіозаписами, про найточніший аналіз та виділення ключових слів, наприклад, він пропонує результати ключових слів і підсумки, а також ділиться ними з іншими користувачами для приміток. Ось чому його в основному використовують під час лекцій, співбесід чи зустрічей, а також для індивідуальних проектів. Otter також може ідентифікувати різні голоси та, отже, призначати різні ідентифікатори динаміків. Одним із обмежень цієї програми є те, що вона підтримує лише

англійську мову, але з цього є переваги у розмірах додатку, вона має вагу всього 102Мб. Порівнюючи з минулим застосунком, це надзвичайно маленький розмір для програмної системи для аналізу аудіофайлів.

Для роботи в програмі потрібно зареєструватись для того щоб усі створені нотатки та аудіозаписи були збережені відповідно до електронної пошти. Була проведена інструкція по додатку. Функціонал з календарем та плануванням було пропущено. Інтерфейс зрозумілий та зручний для користування з смартфону. В безкоштовній версії доступно збереження до 300Мб аудіозаписів.

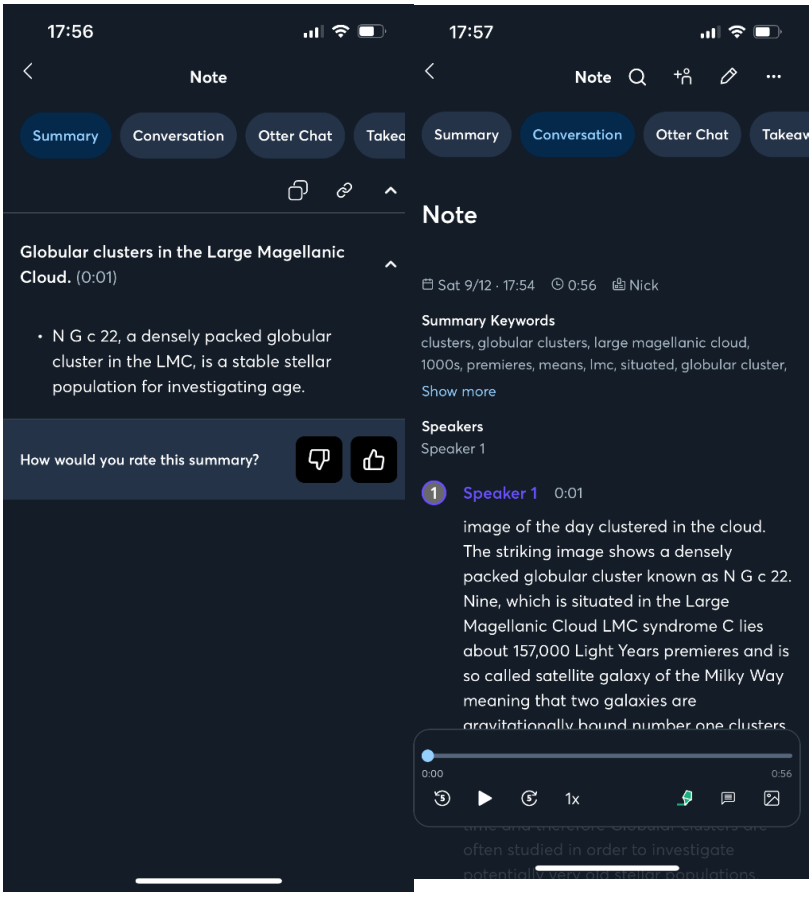

Рис. 1.3.2 Результати тестування та інтерфейс програми Otter

Для тестування були обрані інші дані, оскільки додаток не підтримує завантаження файлу, а тільки саме запис, тому було записано відрізок в 1 хвилину англійського аудіювання. Перетворення слів у текст здійснювалось паралельно з записом. Після завершення запису програма підготовлювала результат, приблизно секунд 5. Результатом став текст з розставленими розділовими знаками, але також зверху на екрані продовжував йти прогрес аналізу тексту. Через секунд 30 екран оновився та загальні дані про текст, де весь текст був коротко описаний в 1 речення

з основними поняттями. Також оновилась сама нотатка, там з’явились ключові слова та розподілення тексту між спікерами.

Приголомшливий аналіз тексту займає немало часу, проте дуже виділяється з-поміж конкурентів. Підсумки та ключові слова визначаються штучними інтелектом, перетворення звуку на текст виконуються скоріше за все методом багатогранної мовної моделі. Аналіз не є алгоритмічним, через витрачений на нього час та велику точність зрозуміло що це виконує штучний інтелект. Аналіз не працює без з’єднання з мережею.

Інтерфейс запису та аналізу звуку є зручним та зрозумілим. Розширяти мовну модель для кінцевого користувача неможливо, програма не передбачає втручання в аналіз, пошуку немає та самі ключові слова програма визначає сама. Не можна змінити записане слово щоб надати фідбек про правильність мовної моделі, таким чином 2 слова з запису довжиною в 1 хвилину були перетворені неправильно.

Octopus PicoVoice

Програмна система представляє собою веб додаток з безкоштовним обмеженим функціоналом та повним функціоналом за підпискою. Очікуваний від програми функціонал був у платній версії, проте наявне безкоштовна демо версія програми лише на 1 запит. Програма може бути використана будь якою операційною системою після реєстрації, оскільки представлена в браузері. Розглянемо її функціонал:

- Аудіозапис та відкриття аудіофайлів
- Перетворення аудіофайлу в текст з його змістом
- Надшвидкий пошук тексту в файлі (платний функціонал)
- Можливе використання в розробці інших програм
- Заявлена швидкість більша за Google Speech-toText та Mozilla Deep speech
- Аналіз вмісту аудіофайлу з паралельним індексуванням (платний функціонал)
- Обробка попри фонові перешкоди

Тестування безкоштовної версії програми відбувалось відрізком в 1 хвилину

дитячої пісні, через заявлений функціонал про можливу роботу з перешкодами, перетворення звуку в текст відбулося надзвичайно швидко, приблизно 5 секунд після завантаження файлу.

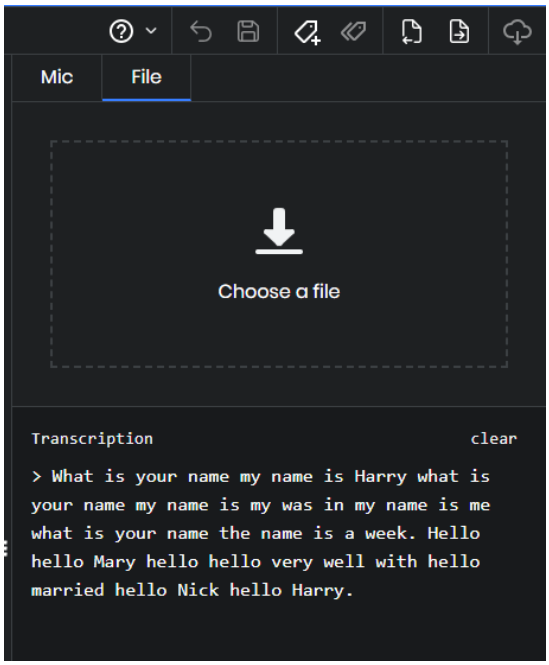

Рис. 1.3.3 Результати тестування та інтерфейс програми PicoVoice

Точної інформації про використані методики аналізу звуку немає, проте виходячи з даних про таку швидкість, можна зробити висновок що використовується комбінований спосіб, а саме акустичне моделювання та нейронні мережі. Нейронні мережі через наявність деяких розділових знаків та виокремлення імен. А акустичне моделювання через надзвичайну точність, проте варто розуміти що такий аналіз дуже важкий у створенні та підтримці та потребує високих наукових досліджень та експериментів. Використані технології дають представлення про можливу швидкість пошуку в аудіофайлах, оскільки проводиться повний аналіз звукової хвилі.

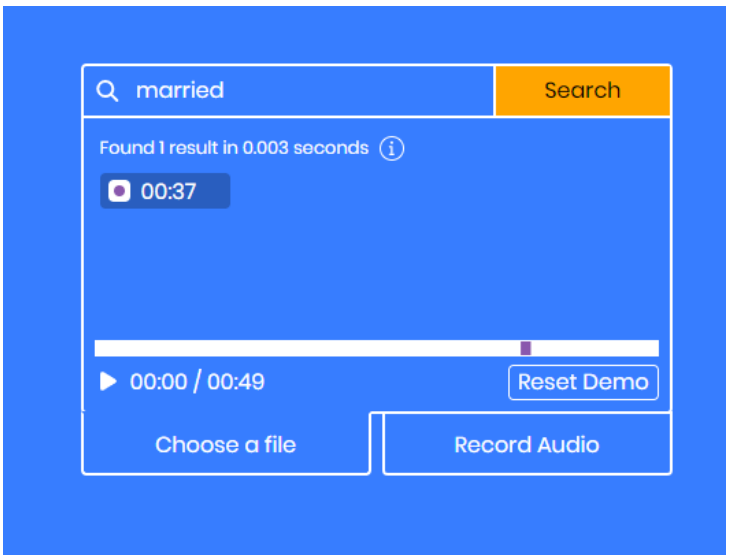

Рис. 1.3.4 Демоверсія програми PicoVoice

Проведене дослідження демоверсії програми показало надзвичайну швидкість пошук. Пошук слова “married” зайняло 0 цілих 3 тисячних секунди за одне входження з 49 секунд файлу. Показаний візуальний проміжок файлу де знаходилось слово, та скільки часу зайняло щоб це слово відтворити, а також точний час відтворення слова, а саме на 37 секунді аудіозапису.

Це єдина програма яка була знайдена в глобальній мережі, яка підтримує пошук співпадінь у аудіофайлах. Програма підтримує завантаження файлів та аудіозапис. Складова аналізу також присутня оскільки показує час та навіть проміжок де ключове слово було знайдено. Контексту немає, проте через визначений проміжок слова, можна переслухати та знайти контекст.

Для узагальнення порівняння програмних системи виведемо його у таблицю 1.1. Оцінка характеристик буде представлена в пояснювальному форматі, а загальна та якісна в числовому форматі по десятибальній шкалі, де 10 – найкраща якість характеристики, а 0 – це відсутність характеристики.

Таблиця 1.1

Порівняння	Braina	Otter	PicoVoice
Розмір програми	340Мб + 2Гб	102Мб	Онлайн
Методика транскрипції	Мовне моделювання	Мовне моделювання	Комбіноване (мовне та

			акустичне)
Якість транскрипції	5/10	9/10	10/10
Швидкість пошуку до об’єму файлу	4:30 до 4:30 (паралельно)	0	0.003 до 0:49
Якість аналізу	5/10	8/10	7/10
Розмір контексту	0/10	7/10	5/10
Наявність звіту	3/10	9/10	8/10
Зручність інтерфейсу	3/10	8/10	8/10
Загальна оцінка	3/10	8/10	9/10

Підсумовуючи дані з таблиці можна зробити короткі висновки з порівняння за його характеристиками. Розмір програми залежить від потреби з’єднання з інтернетом, перша програма не потребувала його, тому мала дуже великий розмір в 2,34Гб. Серед методик транскрипції переважає мовне моделювання, проте за якістю воно може відрізнятись, а у випадку комбінації з акустичним моделюванням показує надзвичайні результати у якості перетворення звуку в текст та швидкості пошуку співпадінь. До якості аналізу входять усі наявні вихідні дані після завершення процесу пошуку чи перетворення аудіоданих. Контекст був представлений лише у 2 програмах та у різному вигляді: у текстовому та візуальному. Візуальний вигляд контексту передбачає його пошук на користувача, тому його оцінка менша. Зручність інтерфейсу була найкраща у веб та мобільних додатків, десктопний додаток був найменш зручним. Загальна оцінка визначила що найкращою програмою для наслідування буде PicoVoice

Висновки

Отже, у цьому розділі були проаналізовані основні поняття та проблема дослідження. Коротко визначені методики та засоби для використання у системі, а також повноцінне дослідження вже наявних додатків зі схожим функціоналом.

Проведено порівняння таких додатків з визначенням переваг та недоліків для

використання цих знань у розробці програмної системи та формування первинних вимог. Також головним завданням було дослідити та проаналізувати таблицю порівняння готових рішень зі схожою тематикою, отже, можна зробити короткі висновки про порівняння характеристик. Розмір програми залежить від необхідності підключення до Інтернету. Наприклад, перша програма, що не потребувала Інтернету, мала великий розмір - 2,34 Гб. Методика мовного моделювання вигідна, але може відрізнятись за якістю. Комбінація з акустичним моделюванням показала вражаючі результати у перетворенні звуку в текст та швидкості пошуку. Якість аналізу включає всі вихідні дані після завершення процесу. Контекст був представлений у двох програмах у текстовому та візуальному вигляді. Візуальний вигляд контексту передбачає його пошук на користувача, але оцінка менша. Найзручніший інтерфейс був у веб-та мобільних додатках, десктопний - найменш зручний. Загальна оцінка визначила PicoVoice як найкращу програму для наслідування.

Після оцінки схожих за тематикою програм можна переходити до визначення вимог. Коротка інформація про існуючі методи вирішення проблеми аналізу вмісту аудіофайлів дозволяє детальніше розкрити алгоритми в наступному розділі.

РОЗДІЛ 2
ХАРАКТЕРИСТИКА ОСНОВНИХ ПІДХОДІВ ДО АНАЛІЗУ ВМІСТУ АУДІОФАЙЛІВ

Встановлення основних методик до аналізу вмісту аудіофайлів дозволяє співставити їх ефективність і придатність для використання в проекті. Тому необхідно поділити підходи на окремі етапи аналізу. Ознака визначена на основі досліджених характеристик складності та універсальності застосування.

2.1 Способи розпізнавання слів в аудіофайлах

Звукова аналітика аудіо файлів є важливою галуззю обробки сигналів, яка включає в себе вивчення та розуміння звукової інформації, що міститься у цих файлах. Аналіз аудіо даних може бути використаний для розпізнавання звуків, розрізнення різних типів звукової інформації, виявлення змін у звуковому середовищі та багато іншого.

Спочатку варто розглянути як розвивались методи обробки звуку.

Еволюція обробки звуку почалась з аналогових методів обробки. Аналогові методи обробки звуку базувалися на використанні електронних апаратів та систем, які опрацьовували звуковий сигнал без його конвертації у цифровий формат. Ці методи стали популярними у минулому, коли технології цифрової обробки звуку ще не були настільки поширеними або доступними. Основні компоненти аналогових систем обробки звуку включали:

- Мікшери – ці пристрої дозволяли змішувати різні звукові доріжки разом, змінюючи гучність кожної окремо. Вони використовувалися для створення багатошарових аудіозаписів, змішування різних інструментальних доріжок або голосів.
- Еквалайзери – такі пристрої дозволяли регулювати частоти звуку. Вони забезпечували можливість підсилення або приглушення окремих частотних діапазонів, таких як низькі, середні або високі частоти, для покращення якості звучання.

- Компресори та лімітери – ці пристрої допомагали контролювати динаміку звукового сигналу. Компресори зменшували динамічний діапазон (різницю між найтихішими та найгучнішими звуками), тоді як лімітери обмежували максимальну гучність сигналу, щоб уникнути перевищення об'єму.
- Ефектори – такі пристрої надавали можливість застосування різноманітних звукових ефектів, таких як ехо, реверберація, фланжери, фазери тощо. Вони змінювали аудіосигнал, додаючи до нього спеціальні акустичні ефекти.
- Аналогові процесори сигналів – ці пристрої мали широкий набір функцій для обробки звуку, включаючи фільтрацію, зміну тембру, модуляцію та інші.

Хоча аналогові методи обробки звуку були популярними і використовувалися в аудіоіндустрії, вони також мали свої обмеження. Наприклад, кожен пристрій міг вносити деякий рівень шуму або спотворення сигналу, а точність регулювання параметрів часто була обмежено. Також, зберігання, обробка та передача аналогового звукового сигналу може бути складнішою порівняно з цифровими методами.

Цифрові технології обробки звуку базуються на використанні цифрових сигналів, що представлені у вигляді чисел. Ці технології стали надзвичайно популярними і широко використовуються в сучасній аудіоіндустрії, оскільки вони дозволяють високу точність обробки звуку та мають численні переваги порівняно з аналоговими методами:

- Точність
- Гнучкість
- Масштабованість
- Стійкість до шуму та спотворень
- Легкість зберігання та передачі

Цифрові технології обробки звуку включають в себе цифрові ефектори, звукові редактори, програмне забезпечення для запису та змішування аудіодоріжок,

синтезатори звуку та інші інструменти, що дозволяють створювати, редагувати та обробляти звукові сигнали. Вони є основою для сучасних музичних студій, кінематографії, телебачення та багатьох інших галузей, де вимагається високий рівень якості та точності обробки звуку.

Наступним кроком еволюції стало семплування та частота дискретизації, ці поняття важкі для розуміння тому слід розглянути їх детальніше. Семплування - це процес перетворення аналогового сигналу в цифровий шляхом збирання точок даних з аналогового сигналу на протязі певного проміжку часу. Сам процес семплування має декілька особливостей. Під час семплування визначається частота дискретизації, тобто кількість семплів, які беруться за одиницю часу. Аналоговий сигнал представляється у вигляді послідовності відбитків (семплів), кожен з яких відображає амплітуду сигналу в конкретний момент часу. Частота дискретизації визначає, як часто збираються ці семпли. Зазвичай це вимірюється в герцах (Гц) або кілогерцах (кГц), наприклад, 44.1 кГц для аудіо CD. Теоремою на основі якої лежить весь функціонал семплування є теорема Котельникова-Шеннона, яка стверджує, що для точного відтворення аналогового сигналу з амплітудою, що не перевищує певної частоти, частота семплування повинна бути принаймні удвічі більшою за максимальну частоту сигналу. На процеси семплування сильно впливає частота дискретизації: зменшення частоти дискретизації може призвести до втрати високочастотних компонентів сигналу, що впливає на якість звуку, а висока частота дискретизації дозволяє зберігати більше деталей аудіосигналу, але також призводить до збільшення обсягу файлу.

Оскільки аудіо файл містить безліч інформації, вагомими факторами аналізу є частотний діапазон, вид звуку, характеристики гучності та часовий проміжок. Розуміння цих факторів дозволяє отримати більш глибоке розуміння аудіо матеріалу. Для аналізу аудіо файлів розвинуто безліч підходів, таких як спектральний аналіз, визначення ритму, розпізнавання мови, розпізнавання звуків за допомогою нейронних мереж та використання синтезу мови. Кожен з цих підходів має свої особливості та використовується залежно від конкретних потреб користувача. Враховуючи складність та універсальність підходів, слід враховувати

особливості роботи з аудіо файлами, що може вплинути на результати та ефективність аналізу. Автоматизована обробка аудіо даних виправдає себе завдяки швидкості та неперервності процесу, що допомагає виявляти та аналізувати звукову інформацію в різних форматах та джерелах з максимальною точністю.

Для початку варто визначити як можна розпізнавати наявність інформації у звуці, для цього використовуються певні методи визначення параметрів аудіосигналів такі як частота дискретизації, бітова глибина, частотний діапазон та спектральний аналіз. Дослідимо їх детальніше.

Частота дискретизації є однією з основних характеристик при роботі з аудіосигналами. Це параметр, який визначає, скільки разів за секунду знімаються або вимірюються зразки аудіосигналу для його подальшої обробки та відтворення. Уявіть аудіосигнал як хвилю, яка має амплітуду (висоту звуку) і часовий характер. Частота дискретизації визначає частоту збереження зразків амплітуди цієї хвилі. Чим вища частота дискретизації, тим більше точок (зразків) збирається за секунду і тим більш точно оцифровується оригінальний аналоговий звук. Частота дискретизації вимірюється в герцах (Гц) і представляє собою кількість зразків на одиницю часу. Наприклад, для стандарту аудіо CD якість звуку задається частотою дискретизації 44,1 кГц. Це означає, що для запису звуку використовується 44 100 зразків на секунду. Важливо, щоб частота дискретизації була достатньою для точного відтворення частот аудіосигналу. Звукові сигнали з високою частотою можуть вимагати більш високу частоту дискретизації, щоб бути відтвореними точно. Якщо частота дискретизації занадто низька, це може призвести до втрати аудіоінформації або спотворень в звуці під час відтворення.

Бітова глибина - це параметр, що визначає кількість бітів, які використовуються для представлення кожного зразка аудіосигналу в цифровому форматі. Цей параметр визначає кількість можливих амплітуд або значень, які можуть бути виражені для кожного зразка звуку. Чим вища бітова глибина, тим більше можливих значень може бути представлено для кожного зразка аудіосигналу, що в свою чергу означає більшу точність і деталізацію цифрового відтворення звуку. Наприклад, стандартний аудіо CD має бітову глибину 16 біт. Це

означає, що для кожного зразка аудіосигналу може бути використано 16 бітів для представлення його амплітуди. Значення бітів можуть бути від 0 до 65,535 (2 в степені 16). Чим більше бітів, тим більша динамічна область і точність відтворення звуку. Використання більшої бітової глибини дозволяє більш точно відтворити динаміку звуку, включаючи тихі і гучні частини, зменшуючи вплив шумів і спотворень на якість аудіоінформації. Однак важливо пам'ятати, що більша бітова глибина призводить до більшого обсягу даних, що використовуються для запису аудіо, і може вимагати більше місця для зберігання.

Частотний діапазон в аудіо визначає ширину діапазону частот, які можуть бути передані або відтворені в аудіосигналі. Він вимірюється в герцах (Гц) і визначає нижню та верхню межі частот, які здатні бути передані або записані в аудіофайлі. Людське вухо зазвичай сприймає звуки у діапазоні від приблизно 20 Гц (нижня межа) до 20 000 Гц (верхня межа). Однак ці межі можуть варіювати в залежності від особливостей слуху кожної людини. У віці або через різні фізіологічні чинники може зменшуватися здатність слухати частоти вище 15 000 Гц або навіть менше. У світі аудіотехніки частотний діапазон може бути визначений як широкосмуговий (від низьких до високих частот) або обмежений з певної причини, наприклад, технічних обмежень пристрою відтворення чи обмежень формату аудіозапису. Високий частотний діапазон дозволяє передавати більше високочастотної інформації, яка може впливати на чіткість і деталізацію звукової інформації, особливо для музичних інструментів або голосу. Широкий діапазон дозволяє відтворювати звук більш вірно і повно. Важливо враховувати, що багато факторів, таких як характеристики обладнання та якість запису, можуть впливати на якість і доступність різних частот у відтвореному аудіосигналі.

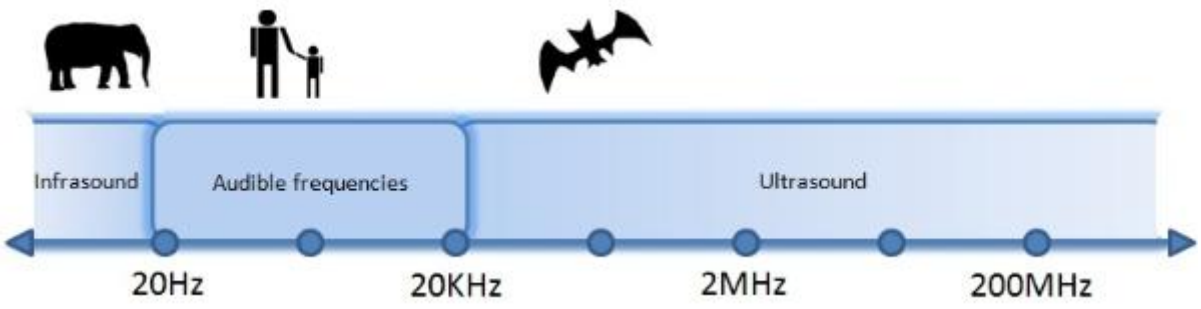

Рис. 2.1.1 Відображення порогу розуміння звуку для людини

Спектральний аналіз - це метод дослідження сигналів, який вивчає їх частотний склад. Цей аналіз дозволяє розкласти сигнал на його складові частоти, що дозволяє визначити, які конкретні частоти складають сигнал і яка їх інтенсивність. Основним інструментом спектрального аналізу є перетворення Фур'є. Це математичний метод, який розкладає сигнал у частотному просторі. Результатом цього аналізу є графік, який показує, які частоти присутні в сигналі і як їхня сила змінюється з часом. Спектральний аналіз широко використовується у багатьох галузях, зокрема у звукозапису, радіоінженерії, обробці сигналів та науці про матеріали. В аудіо, наприклад, спектральний аналіз дозволяє аудіоінженерам аналізувати та впливати на різні частоти звуку, що допомагає в покращенні якості звукового сигналу. В загальному, спектральний аналіз надає можливість розглядати сигнали з точки зору їх складових частот, що є важливим для багатьох областей науки та техніки.

До того ж окремо існують вже готові методи розпізнавання слів в аудіофайлах, які використовують наведені вище методи обробки звуку. Комбінація деяких методів разом чи виділення головних та найефективніших властивостей кожного з них привело до формування окремих методик:

- Акустичне моделювання
- Нейронні мережі
- Використання синтезу мови
- Комбінований підхід

Акустичне моделювання як підхід розпізнавання мовлення використовує статистичні моделі, які працюють з акустичними характеристиками звукових сигналів. Акустична модель може використовувати характеристики звуку, такі як спектральний аналіз, мел-кепстральні коефіцієнти та інші функції для виявлення слів у звуковому сигналі. Аудіодані обробляються для підготовки до подальшого аналізу. Це може включати фільтрацію шумів, нормалізацію гучності, розбиття аудіофайлів на частини або видалення непотрібних компонентів. Наступний етап включає в себе вибір та екстракцію важливих акустичних ознак з аудіоданих.

Ознаки можуть включати часові та частотні параметри, такі як спектральні коефіцієнти, частоти формант, енергію сигналу та інші. І потім на основі отриманих ознак створюються акустичні моделі для різних фонем, слов або фраз. Ці моделі можуть бути статистичними (наприклад, гаусівські змішані моделі) або засновані на глибокому навчанні (нейронні мережі). Акустичне моделювання для розпізнавання мовлення постійно вдосконалюється завдяки новим технологіям аналізу звуку, розвитку алгоритмів машинного навчання та покращенню якості акустичних моделей, що призводить до підвищення якості розпізнавання мовлення

Застосування нейронних мереж, зокрема рекурентних нейронних мереж (RNN – recurrent neural networks), довга короткочасна пам'ять (LSTM – long short time memory), використовується для розпізнавання слів, проте має складне навчання. Ці моделі можуть вчитися на великому обсязі даних і тренуватися на аудіозаписах для передбачення слів або фонових перешкод. Мережі нейронів в цьому випадку отримують на вхід аудіодані, такі як розмови або звукові записи. Вони складаються з багатьох шарів, де кожен шар містить нейрони. Нейрони у цих мережах обробляють вхідні дані, встановлюючи зв'язки між різними частинами аудіосигналів. Під час тренування нейронні мережі використовуються для пошуку взорів у великих наборах аудіоданих. Вони аналізують різні акустичні особливості, такі як частоти звуків, їхню інтенсивність та звукові характеристики, щоб вивчити особливості різних звуків і розмовних мовленнєвих моделей. Після тренування ці мережі стають здатними розпізнавати мовлення у реальному часі. Вони аналізують вхідні аудіодані та співставляють їх з раніше навченими шаблонами або категоріями мовлення, щоб визначити, що саме було сказано або виражено у вигляді тексту. Застосування нейронних мереж для розпізнавання мовлення дозволяє покращити точність і швидкість розпізнавання, а також може адаптуватися до різних голосів, акцентів та акустичних умов для досягнення кращих результатів.

Використання синтезу мови іноді може бути корисно де штучно згенеровані голоси використовуються для порівняння з аудіозаписами та ідентифікації слів. Синтез мови використовується для створення аудіосигналів, які відтворюють звук мовлення за допомогою комп'ютерних програм або штучних систем. У контексті

розпізнавання мовлення, синтез мови використовується для створення аудіосигналів з текстової інформації. Процес синтезу мови починається з текстового вводу. Комп'ютерна програма або система отримує текст, який потрібно промовити. Текст може бути поданий у вигляді фрази, речення або навіть окремих слів. Системи синтезу мови використовують різні алгоритми та моделі генерації звуків. Одним з найпоширеніших підходів є використання природних голосів або моделювання звуків людського мовлення. Це може включати в себе записи людських голосів та їх подальше аналіз для відтворення відповідного звуку. Ключовими етапами синтезу мови є вибір відповідних звуків для кожної фонеми (малих звукових одиниць мови), їхнє поєднання для створення слова та речення, а також надання натурального тону та інтонації для звучання природно. Цей підхід дозволяє створювати аудіосигнали, які відтворюють мовлення, що було введене у текстовій формі. Коли такий синтезований аудіосигнал відтворюється, його можна використовувати для аналізу мовлення за допомогою алгоритмів розпізнавання мовлення, які обробляють отримані звукові дані для визначення тексту або мовлення.

Комбінований підхід передбачає що іноді комбінація різних методів може дати кращі результати. Наприклад, можна поєднувати акустичні моделі з нейронними мережами для покращення точності розпізнавання. Така методика є більш абстрактною, тому що не має визначених комбінацій, будь яка кількість комбінацій перечислених підходів призведе до застосування комбінованої методики в плані експерименту для визначення підходящих варіантів.

Наведені методики розпізнавання слів у аудіофайлах мають відмінні один від одного переваги, тому використання кожного з них має мати окреме дослідження доцільності та ефективності.

2.2 Машинне навчання для розпізнавання мови

Машинне навчання для аналізу мови – це окрема галузь штучного інтелекту, яка запроваджує та використовує та алгоритми машинного навчання для розуміння, інтерпретації та обробки природної мови у цифровому записі.

Мовне навчання включає в себе розробку статистичних моделей для аналізу та розуміння людської мови, засновану на розподільчих властивостях послідовностей слів. Цей процес передбачає значний обсяг навчання на великих об'ємах текстових даних, для точного та надійного відображення мови.

Мовна модель - це статистична модель, яка використовується для оцінювання ймовірності послідовності слів у мові. Її основна мета - передбачити наступне слово або символ на основі попереднього контексту. Мовні моделі допомагають вирішувати завдання, пов'язані з розпізнаванням мови, машинним перекладом, генерацією тексту та іншими завданнями обробки природної мови.

По-перше визначимо типи мовних моделей.

Уніграмна модель можна розглядати як поєднання кількох одноставних скінченних автоматів. Вона визначає ймовірності різних слів чи термінів у контексті. У цій моделі ймовірність кожного слова залежить лише від ймовірності цього слова в документі. Таким чином, кожен вузол відповідає одноставному скінченному автоматові. Цей автомат має розподіл ймовірностей для всього словника цієї моделі, який складає 1. Різні документи мають уніграмні моделі з різними ймовірностями вхідних слів. Для формування ймовірностей вхідних слів для кожного запиту використовуються розподіли ймовірностей з різних документів.

Двосторонні представлення визначаються як контекст перед та після у всіх шарах. Крім того, без маркера кінця речення ймовірність неграматичної послідовності завжди буде вищою, ніж ймовірність довшого речення.

Нейромережні моделі мов, також називаються неперервно-просторовими моделями мов використовують неперервні представлення або вкладення слів для передбачення. Ці моделі використовують нейронні мережі.

Неперервно-просторові вкладення допомагають пом'якшити проблему розмірності при моделюванні мови. Зі збільшенням розміру тренованих моделей мов на більшому обсязі текстів зростає кількість унікальних слів. Кількість можливих послідовностей слів росте по експоненті з розміром словника, що призводить до проблеми розрідженості даних через експоненційно велику кількість послідовностей. Тому для кращої оцінки ймовірностей бажано мати статистику, а

для правильної – потрібно.

Та найпоширенішою формою мовної моделі є n-грамна модель, де n вказує на кількість попередніх слов або токенів, на основі яких розраховується ймовірність наступного слова. N-грами - це послідовності підряд розташованих n елементів, які спостерігаються в корпусі тексту. Наприклад, в n-грамній моделі з n=2, ймовірність наступного слова визначається на основі попереднього слова. Така модель може розраховувати ймовірність появи окремого слова в реченні і використовувати її для передбачення наступного слова. Мовні моделі можуть бути побудовані за допомогою різних алгоритмів, таких як наївний Баєс, марковські моделі, нейронні мережі та інші. Ці моделі вимагають тренування на великому наборі текстових даних, щоб зрозуміти тексти мови та встановити ймовірності для наступних слів.

Такі моделі є складними, проте вони допоможуть не тільки в розпізнаванні мови, а й у пошуку співпадінь та дослідженні контексту.

Застосування мовних моделей для розпізнавання мови передбачає використання як розмовних, так і лінгвістичних підходів. Мовне навчання є потужним та ефективним способом аналізу та розуміння мови і, завдяки постійній роботі та дослідженням у цій галузі, воно знаходить все більше застосувань у сфері обробки природної мови.

Для того, щоб використовувати методи мовного моделювання, потрібно виконати наступні кроки:

Збір та підготовка даних: отримання великого набору текстових даних, які будуть використовуватись для навчання моделі. Дані можна зібрати самостійно або скористатись існуючими наборами даних.

- Токенізація: розбиття текстових даних на окремі слова або токени. Це може включати поділ на окремі слова, символи або навіть рядки.
- Побудова моделі: використовуючи зібрані дані та токенізацію, необхідно побудувати мовну модель. Існує багато різних архітектурних моделей, таких як рекурентні нейронні мережі, довга короткочасна пам'ять чи трансформерні моделі, які можна

використовувати для цієї мети.

- Тренування моделі: з використанням зібраних даних та побудованої моделі провести процес тренування. Це означає передачу текстових даних моделі, щоб вона "навчилась" відтворювати текст та розмірковувати на основі контексту.
- Оцінка моделі: після тренування моделі потрібно оцінити її ефективність, наприклад, використовуючи метрики, які враховують точність передбачення моделі або можливість зрозуміти та згенерувати значущий текст.
- Використання моделі: Одержавши навчену модель, ви можете використовувати її для генерації тексту на основі заданого контексту або для розпізнавання мови. Для цього можна використовувати API або інтерфейси програмування для виклику моделі і отримання результатів.

Загалом, мовне моделювання - це складний процес, який вимагає підготовки даних, побудови та тренування моделі, а також оцінки та використання результатів. Процедури та методи можуть різнитися в залежності від використовуваних алгоритмів та задачі, на які спрямована модель.

Також є різноманітні алгоритми навчання моделей. Розберемо основні з них.

Класифікація – це процес призначення об'єктів до категорій або класів на основі їхніх характеристик. В машинному навчанні це означає навчання моделі розпізнавати певні патерни або властивості вхідних даних і віднесення нових об'єктів до вже вивчених класів. Наприклад, у класифікації тексту модель може визначати, чи є певний текст позитивним, негативним або нейтральним, виходячи з його вмісту та структури. Класифікація широко застосовується в області розпізнавання образів, аналізу текстів, медицинських діагнозів та багатьох інших галузях для розподілу об'єктів до різних категорій або класів.

Регресія – це метод в машинному навчанні, який використовується для прогнозування числових значень на основі залежностей між змінними. Основна мета регресії - це з'ясувати, як одна змінна впливає на іншу. У випадку одновимірної

регресії ми розглядаємо залежність між однією прогностичною (незалежною) змінною та однією змінною відгуку (залежною). У багатовимірній регресії маємо кілька прогностичних змінних та одну змінну відгуку. Модель регресії шукає функціональні зв'язки між цими змінними, намагаючись побудувати лінійні або нелінійні відношення між ними. Результатом регресійного аналізу є прогнозовані значення змінної відгуку на основі значень прогностичних змінних.

Кластеризація – це метод машинного навчання, що використовується для групування схожих об'єктів в класи або кластери. Мета полягає у розділенні набору даних на підмножини, де об'єкти всередині кожного кластера схожі між собою, а між різними кластерами - відмінні. Алгоритми кластеризації приймають вхідні дані та групують їх на основі подібності або відстані між ними. Це може бути основано на різницях у властивостях об'єктів або їх характеристик. Зазвичай, кластеризація дозволяє виявити природні групи в даних, які можуть бути невидимими при перегляді вихідних даних. Цей метод застосовується в багатьох областях, таких як обробка даних, біоінформатика, аналіз тексту, медицина, сегментація зображень та інші, для виявлення внутрішніх структур в наборах даних і отримання нових інсайтів з великих обсягів інформації.

Також є інші методи, вони рідше використовуються, проте їх огляд є важливим також. Дерева рішень та випадковий ліс – дерева рішень використовуються для прийняття рішень на основі правил, зазвичай для класифікації. Випадковий ліс - це ансамбль дерев рішень, які працюють разом для досягнення кращих результатів. Навчання без учителя – до цієї категорії належать алгоритми кластеризації та вимірювання подібності. Вони дозволяють використовувати дані без попередньо визначених міток чи цільових значень. Підкріплене навчання – ці алгоритми вчаться на основі взаємодії з навколишнім середовищем, отримуючи відповідь на свої дії. Наприклад, Q-навчання, яке використовується в робототехніці та іграх.

2.3 Огляд основних алгоритмів пошуку ключових слів

Одним з основних видів алгоритмів є пошук і має немало різних вирішень. Деякі з цих методів ефективні лише для певних об’єктів пошуку в визначених середовищах. Тому ключовим є вирішення проблеми пошуку довільних аудіоформатів. Отже, розглянемо найефективніші методи. Для спрощення складного тесту використовуємо скорочення: n — довжина тексту, а m — довжина ключового слова.

Алгоритм Боєра-Мура — це швидкий алгоритм для пошуку підрядків у текстах, який може бути дуже ефективним для великих обсягів даних, навіть більше, алгоритм має особливість що його ефективність буде зростати разом з розміром тексту. Основна ідея полягає в тому, щоб під час пошуку використовувати правило «швидкого споживання».

Отже, визначимо шлях для пошуку співпадінь в тексті для цього алгоритму. Першим етапом буде підготовка таблиці зсувів, де для кожного символу в шуканому слові визначається максимальний можливий зсув. Починаючи з кінця шуканого слова, алгоритм порівнює його з відповідною позицією в тексті. При знайденні невідповідності визначається новий зсув, використовуючи таблицю зсувів. Алгоритм визначає на скільки символів можна зсунутися в тексті, використовуючи таблицю зсувів. Це дозволяє пропустити порівняння, якщо останній символ шуканого слова не відповідає тексту. Потім відбувається, зсув у право, при кожній невідповідності алгоритм використовує таблицю зсувів для визначення величини зсуву у право.

Алгоритм Бойера-Мура особливо ефективний, коли шукане слово досить довге або коли перевіряється велика кількість тексту на входження шуканого слова. Це дозволяє швидко переходити через тексти, максимально уникати порівнянь, що робить його дуже ефективним для практичного застосування.

Алгоритм Кнута-Морріса-Пратта – це швидкий алгоритм пошуку підрядка в тексті, який також ґрунтується на ефективному використанні префікс-функцій. Основна його ідея полягає в тому, щоб уникнути повторних порівнянь символів, які

вже були порівняні.

Тож опишемо основні кроки алгоритму. Першим кроком є побудова префікс-функції для шуканого підрядка чи як його прийнято називати в цьому алгоритмі – паттерн. Префікс-функція для кожної позиції паттерну вказує на найбільшу можливу довжину суфікса, який є також його префіксом. Починаємо порівнювати паттерн і текст, виконуючи зсув залежно від відповідності символів. Якщо символи не збігаються, алгоритм використовує префікс-функцію для визначення наступного місця, з якого можна продовжити порівняння, пропускаючи збігаються підстроки. Коли виявляється неспівпадіння, алгоритм використовує префікс-функцію для визначення нової позиції щоб порівняти з минулою, дозволяючи пропустити порівняння зі схожими фрагментами тексту, які вже були зіставлені з паттерном.

Алгоритм Кнута-Морріса-Пратта є ефективним у використанні пам'яті, оскільки він не потребує додаткового простору, окрім пам'яті, необхідної для зберігання префікс-функції. Він також ефективний виключно через оптимізацію порівнянь, що дозволяє прискорити пошук, уникнувши повторних порівнянь, що вже були виконані.

Алгоритм Рабіна-Карпа – це алгоритм пошуку ключового слова в тексті, який використовує метод хешування для ефективного пошуку. Основна ідея полягає в порівнянні хеш-значень шуканого слова та вікна тексту, що рухається вздовж тексту.

Розглянемо основні кроки алгоритму Рабіна-Карпа. Починаємо з обчислення хеш-значення для ключового слова та першого вікна тексту. Хеш-значення використовується для швидкого порівняння підрядка з текстом. Порівнюємо його хеш з хешем першого вікна тексту. Якщо хеші співпадають, виконується додаткова перевірка символів підрядка і тексту для визначення точного збігу. А якщо хеші не співпадають, вікно тексту зсувається на один символ вперед і обчислюється нове хеш-значення для нового вікна тексту. Цей процес триває до знаходження збігу ключового слова з вікном тексту або до кінця тексту.

Алгоритм Рабіна-Карпа дозволяє швидко здійснювати пошук, оскільки він використовує хеш-значення для порівнянь. Однак важливою є правильність

обчислення хешу, оскільки можливі колізії (ситуації, коли різні послідовності символів мають однакове хеш-значення), що може призвести до помилкового визначення збігу. Також, при обробці великих текстів, можуть виникнути проблеми з ефективністю через необхідність постійно обчислювати хеші для великої кількості підрядків і тексту, що викликає сильне навантаження на процесор і дуже залежить від кількості ядер процесора.

Алгоритми статистичного аналізу використовуються для виявлення шаблонів, зв’язків або регулярних випадків у великих обсягах даних. Використовується в основному нейронними мережами та разом з мовними моделями. Ключова мета – це здійснити аналіз, виявити закономірності та зробити прогнози на основі статистичних методів. Таких алгоритмів є декілька, проте ціль у них одна, прогнозувати наступні слова, розглянемо декілька з них.

Кластерний аналіз – визначає групи (кластери) у великому наборі даних, де схожі об'єкти або спостереження згруповані разом. Мета - знайти приховані відносини між даними і відокремити їх на основі спільних ознак.

Факторний аналіз – використовується для виявлення зв’язків між різними змінними у великому наборі даних. Цей метод дозволяє з'ясувати, які змінні пов'язані між собою і які фактори впливають на певні явища чи процеси.

Регресійний аналіз – використовується для встановлення взаємозв’язку між залежною змінною і однією або кількома незалежними змінними. Цей аналіз дозволяє розуміти, як змінюється одна змінна при зміні іншої.

Аналіз часових рядів – використовується для дослідження даних, зібраних відповідно до часу. Цей аналіз допомагає виявити тенденції, цикли, сезонні зміни та інші регулярності у часових даних.

Дискримінантний аналіз – використовується для відокремлення двох чи більше класів об'єктів чи спостережень на основі декількох змінних чи ознак.

Підсумовуючи визначені алгоритми пошуку слова у тексті можна зробити висновок що вони мають свою окрему особливість, якийсь для великих об’ємів тексту, якийсь для багатьох входжень ключового слова. Проте передбачити такі випадки важко чи взагалі неможливо, тому для кращої ефективності використати

переваги кожного з них та спробувати згуртувати ці методи разом. Дослідження алгоритмів статистичного аналізу показало можливо надзвичайно плідну співпрацю з методом розпізнання мовлення в аудіофайлах за допомогою мовних моделей.

2.4 Аналіз факторів що впливають на якість дослідження контексту

Контекст – це частина певного зв’язаного чи не зв’язаного тексту яка може мати закінчену думку та при цьому дозволяє точно встановити значення слова чи словосполучення. Однак, при пошуку ключового слова в нашому випадку, контекст вважається словами, які знаходяться поруч з шуканим словом. Мета такого контексту не тільки розуміти суть ключового слова, але й надати більше інформації про нього.

Існують різні підходи до використання контексту, вони відрізняються за кількістю слів, напрямком, граматичними конструкціям та знаками пунктуації. За дослідженнями, найкращим контекстом при пошуку числових або вирішальних значень ключового слова є наступне або попереднє слово. Наприклад, при пошуку слова "сума", наступним словом часто буде значення цієї суми. З іншого боку, при пошуку слова "гривень", значення суми може бути попереднім словом. Такий вибір контексту є ефективним для уточнення числа. Однак, у наведеному прикладі може бути ситуація, коли у тексті є інше уточнення, яке не є шуканим. Тому можна використовувати розмір контексту, який складається з двох слів з кожного боку.

Якщо текст більш повний і структурований, останні два слова будуть найцікавішими об’єктами знайденого контексту, так як вони можуть містити значення і мету використання ключового слова. Ще одне поширене припущення щодо розміру контексту є використання 3 слів в кожному напрямку.

Цей підхід найбільш ефективний для пошуку контексту в художній літературі. Так як в літературних текстах або текстах, написаних без прив’язки до категорії зазвичай використовуються прикметники, то доступ до шуканого значення може бути довшим за якийсь науковий твір. Наведемо приклад в підтвердження думки, ключове словосполучення «Перший Код», контекст попередніх слів буде складати

«осіння новинка кіноіндустрії», а наступних – «прославила українських програмістів». З даного прикладу додаткове розуміння складає третє слово попереднього контексту про можливого автора чи суб'єкта фільму, що має ключове значення та з іншого напряму приблизну дату виходу фільму. А ті слова, які знаходились ближче до шуканого слова є менш корисними у пошуку значення.

Після наведених прикладів може скластись думка що для контексту потрібно якомога більше слів, але це не так, тоді втрачається суть знаходження контексту програмою, оскільки все рівно його має аналізувати людина та обирати більш ефективні самостійно. Звісно для кожного типу тексту розмір контексту є по-своєму важливим. Досліди статистики можуть бути не такими корисними у різних випадках, але я вважаю що найкращим рішенням буде надати можливість користувачеві самому змінити цей розмір, але початковим значенням буде середина. «Золотою серединою» з-поміж різних варіацій вважається найкращими рішенням, оскільки цієї інформації буде достатньо для поверхневого розуміння контексту, а при можливій потребі знаходження більш детального контексту, повторний пошук вже знайденого фрагменту буде легшим для сприйняття та більш ефективним для програми.

Критерії пошуку залежать від категорії тексту й можуть бути більш точними, коли враховується додатковий напрямок пошуку. Якщо ми зважаємо на статистику, то більш ефективнішим буде шукати наступне слово в контексті, тобто у правому напрямку. Дослідження підтверджують цей підхід, проте, оскільки метою є збільшення шансу знаходження ключового слова, має сенс використовувати обидва напрямки пошуку. Звичайно, пріоритет надається наступному слову. Тому, якщо контекст має великий розмір і стає незручним для користувача, варто зосередитись на словах, що знаходяться праворуч від знайденого ключового слова.

Наступні фактори рідко доступні саме у аналізі мовлення певного тексту, проте така можливість є, тому її варто також розглянути.

Наявність або відсутність розділових знаків здатна менш покращувати ефективність, але може бути корисною для уточнення. Розділові знаки поліпшують розуміння знайденого ключового слова для людини, оскільки дозволяють

визначити, до якої граматичної конструкції воно належить або у якому контексті воно вживається, якщо воно виділене розділовими знаками. Однак, розділові знаки не повинні погіршувати ефективність дослідження контексту, що може статись, якщо розділові знаки замінять слова з контексту. Кращим рішенням буде розгляд контексту так, якщо слово в ньому може містити розділові знаки. Таким чином, ефективність дослідження буде вищою, а контекст буде доповнюватися граматичною інформацією для зручнішої обробки.

Граматичні конструкції такі як речення чи абзаци впливають на поєднання контексту походять від минулого підходу, оскільки розділяються розділовими знаками. Речення є окремою та відносно незалежною думкою в тексті і може складатись багатьох слів, рідко одного, але таке також може бути, наприклад вигук. За допомогою цього визначення висновком буде що контекстом може бути все речення або навіть складатися з багатьох речень, тому поєднання буде бажаним. А абзац може складатись з одного чи декількох речень зі зв’язком їх за сенсом. Тобто ця велика конструкція передбачає, що речення одного абзацу має менше спорідненості, аніж сусідні реченням. Тоді абзац не варто поєднувати в контексті, таке поєднання має в наслідку розходження сенсу, оскільки в різних місцях контексту буде різний зміст.

2.5 Вимоги до спроектованої програмної системи

Перед проектуванням зазвичай визначають вимоги, тобто певну специфікацію програмної системи. Вони поділяються за функціональністю тобто, функціональні та нефункціональні. Функціональні вимоги повинні включати опис бажаної логіки системи для перетворення вхідних до вихідних даних. Також вони визначають роботу системи, включаючи складові алгоритмічні та ефективності виконання. Нефункціональні вимоги описують представлення системи користувачеві а саме такі критерії: вимоги до безпеки, надійності, оновлень, зручності та лаконічності.

З урахуванням висновків з попередніх розділів можна скласти основну низку функціональних вимог для аналізу вмісту аудіофайлів. Визначено, що оптимальним

рішенням буде використання різних методів в їх найефективніших середовищах. Однозначно для роботи з обробкою аудіофайлу будуть використовуватись мовні моделі через велику складність повного аналізу та потрібної певної наукової складової в цій сфері. Для цього потрібно виконати налаштування та навчання штучного інтелекту на розпізнавання слів у звуковій доріжці аудіофайлу. Для більш точного відсіювання слів з можливим пошуком співпадінь на шукане слово варто застосувати алгоритм Рабіна-Карпа. Алгоритм Рабіна-Карпа використовується для точного пошуку за хеш-значенням, тому при налаштуванні отримувати результат при обробці аудіофайлу в текст моментально, порівняння хеш-сум не має негативно впливати на швидкість аналізу вмісту файлу.

З вивченням факторів, які впливають на аналіз контексту, виокремлюються основні критерії структуризації цього контексту. Найоптимальнішим розміром контексту вважається середня кількість слів, яка складає 2 слова. При цьому перевага надається наступним словам, проте рекомендується використовувати обидва для більшої точності. Використання розділових знаків важливе, проте не має призвести до зменшення обсягу контексту, але повинно бути включене у слова чи поєднане з контекстом іншого речення, уникнувши абзаців, оскільки це може вплинути на зміст контексту. Такий підхід забезпечує максимальну ефективність виявленого контексту у більшості випадків, проте це стандартні параметри. Варіативність їх налаштувань робить підхід до використання більш гнучким.

До функціональних вимог також варто віднести тип файлів, які можливо обробляти та приймати до системи. Типи файлів які потенційно можуть мати звукову доріжку багато, серед них окремо звукових форматів: MP3, WAV, AIFF, FLAC, OGG, MIDI, WMA, AAC, AMR, та відеоформатів: AVI, MP4, MKV, MOV, WMV, FLV, WebM, 3GP, MPEG, VOB. Але серед них є один особливий формат – WAV. Формат WAV (Waveform Audio File Format) - це формат зберігання аудіоданих без стиснення, що зберігається без втрат. Основна його перевага – це те що такий формат зберігає звук без будь-якого стиснення, що означає, що весь оригінальний звук зберігається, не втрачаючи якості. Це важливо для обробки звуку мовлення, оскільки зберігається висока якість звуку, важлива для точного аналізу

мовлення. Формат WAV не стискає дані, тому забезпечує відмінну якість аудіо, особливо для аналізу різних аспектів мовлення, таких як інтонація, темп, акцент та інші нюанси. До того ж підтримує якісний стереозвук, що може бути корисним при аналізі акустичних особливостей мовлення. А також цей формат є відкритим стандартом, тому його легко використовувати та обробляти в різних програмах для аналізу та обробки звуку. Варто відзначити що будь який формат можна конвертувати в WAV зі збереження максимально наявної якості звуку з файлу що конвертується.

Ці переваги роблять WAV ідеальним форматом для збереження та обробки звуку мовлення, особливо при потребі точності та відтворення всіх аспектів аудіоданих. Але також варто перейти до недоліків: з боку використання пам’яті його ефективність не є найкращою, тому величина файлу може бути більшою з порівнюваних.

Перейдемо до функціональних вимог, фундаментом для них стануть програмні та апаратні вимоги. Розробка програмної системи має відбуватися на мовах програмування Kotlin Ktor для сервісної частини та React TypeScript для клієнтської. Підтримка основних операційних систем можлива завдяки підтримці мовами та осередку кроспллатформенності. Потрібні бібліотеки що використовуються у проекті будуть зкомпільовані та додані до загальної зборки додатку для уникнення зайвого встановлення для користувача. А також для використання на різних платформах обрано і середовище роботи програмної системи, а саме веб-браузер, який за замовчуванням є в більшості з них, а в інших з легким встановлення. Робота системи в веб-браузері зовсім не потребує інтернет підключення, а також дозволяє налаштування окремого серверу та створення хосту для підключення з інших комп’ютерів.

Розглянемо детальніше переваги мови Kotlin, це мова програмування, що має кілька переваг для розробки програми аналізу аудіофайлів. Вона є простою для вивчення і зрозуміння, що полегшує роботу розробників, особливо тих, хто тільки починає свій шлях у програмуванні. Kotlin підтримує функціональне та об'єктно-орієнтоване програмування, що дозволяє розробникам використовувати різні

підходи для створення програми. Однією з головних переваг Kotlin є його безпека. Він пропонує безпечну систему типів, яка допомагає уникнути багатьох типових помилок під час написання коду, таких як помилки при приведенні типів або доступ до недійсних об'єктів. Це сприяє стабільності та надійності програми. Kotlin також має простий синтаксис, що робить код більш зрозумілим та зменшує ймовірність виникнення помилок. Ця мова дозволяє писати менше коду для досягнення тих самих результатів, порівняно з іншими мовами програмування, що спрощує його розробку та підтримку. Kotlin є мультиплатформеною мовою, тобто код, написаний на Kotlin, можна використовувати на різних платформах, що полегшує розробку та підтримку програми для аналізу аудіофайлів на різних пристроях або операційних системах.

Тепер поговоримо про доцільність використання комбінації React та TypeScript для фронтенду.

Typescript - це мова програмування, яка є розширенням JavaScript. Основна її особливість - це типізація, яка дозволяє визначати типи даних для змінних, параметрів функцій та інших об'єктів. Це полегшує роботу розробників, допомагає уникнути помилок та покращує розуміння коду. Typescript дозволяє використовувати сучасні можливості JavaScript, одночасно забезпечуючи підтримку для більш високого рівня безпеки та стабільності коду.

React - це бібліотека JavaScript для створення користувацьких інтерфейсів, яка дозволяє побудувати ефективні веб-додатки з допомогою компонентного підходу. Основні переваги React включають в себе використання віртуального DOM, що полегшує оновлення сторінок та забезпечує оптимальну швидкість роботи додатків. Компоненти React дозволяють створювати елементи які можна використовувати повторно, що спрощує розробку та підтримку коду.

Використання Typescript разом з React дозволяє збільшити ясність та безпеку коду в процесі розробки веб-додатків. Під час побудови веб-додатку Typescript допомагає визначити типи для компонентів, подій та даних, тим самим спрощуючи взаємодію з React. Комбінація цих технологій дозволяє побудувати додаток з чистим та структурованим кодом, забезпечуючи високу швидкодію та зручність розробки.

Серед апаратних вимог найбільше уваги потребує основний процес обробки звуку, оскільки він найбільше потребує вираховувальних потужностей. Мінімальні характеристики для роботи програми включають процесори з тактовою частотою більше 2 ГГц, бажано випущені після 2018 року, обсяг оперативної пам'яті - 8 ГБ, і наявність вільного простору на жорсткому диску - 100 МБ. Однак для оптимальної ефективності програми рекомендується використання процесорів від 2.5 ГГц та обсягу оперативної пам'яті більше 8 ГБ. Але особливо потрібна наявність звукової карти, її новизна та якість напряму впливає на аналіз та обробку звуку.

Ще одним ключовим моментом для ефективності роботи є тип та характеристики жорсткого диску, оскільки програма працює безпосередньо з пам'яттю та при високому навантаженні файлами з великим її обсягом. Незважаючи на те, що програма може працювати з будь-яким типом диска, слід зауважити, що HDD диски працюватимуть повільніше, у порівнянні з SSD, а останні, в свою чергу, будуть менш продуктивні за M2. Частота диску також важлива - чим вище, тим краще. Ці відмінності фундаментальні для роботи комп'ютера і впливають на швидкість обробки даних з аудіофайлів, що потребує визначення під час формування вимог до програмної системи.

Наступна категорія вимог - це надійність і безпека. Конфіденційність аналізу аудіофайлів - це питання, пов'язане з збереженням та захистом конфіденційної інформації, яка може бути виявлена або отримана під час аналізу аудіофайлів. При обробці аудіоданих важливо забезпечити захист особистої інформації, такої як особисті розмови, конфіденційна інформація про користувачів чи будь-яка інша приватна інформація, що може міститися в аудіофайлах. Забезпечення конфіденційності аналізу аудіофайлів включає в себе ряд заходів безпеки та політик, спрямованих на обмеження доступу до особистої інформації, шифрування даних під час їх передачі та збереження, яке дозволяють аналізувати дані без ризику розголошення особистих даних. З урахуванням важливості приватності та конфіденційності, дослідники та розробники працюють над технічними рішеннями та етичними стандартами для забезпечення безпеки особистих даних під час аналізу аудіофайлів.

Конфіденційність під час аналізу аудіофайлів штучним інтелектом можна забезпечити кількома способами. Один із них - це застосування шифрування для збереження та передачі даних у зашифрованому вигляді. Також важливо враховувати анонімізацію особистої інформації, щоб забезпечити безпеку особистих даних, які можуть бути в аудіофайлах. Технології обробки даних можуть використовувати методи, що відділяють особисту ідентифікаційну інформацію від загальних даних, що дозволяє зберігати та аналізувати дані без ризику порушення конфіденційності користувачів.

Усі випадки контролюються операційною системою. Якщо файл недоступний, зашифрований або захищений паролем, програми не можуть отримати до нього доступ, доки не буде вирішено відповідно до можливостей операційної системи та доступу користувача. Програма надійна, оскільки використовує найновішу та найпопулярнішу мову програмування, а винятки в низькорівневому доступі до вмісту аудіофайлів уже виправлені, і їх виникнення дуже мало. Захист даних також слід включити до цих категорій. Програма є повністю конфіденційною, а також програма відокремлена від функцій телеметрії та передачі даних, тому вся відповідальність лежить на користувачеві.

Якщо коротко, вимоги до інтерфейсу – зручність і простота. спочатку основним важливим елементом було визначення функціональних вимог для стандартних варіантів виконання аналізу, оскільки основним недоліком програм аналогів, розглянутих у першому розділі, було високе навантаження на інтерфейс. Тому варто вводити в систему мінімум вхідних даних. Звичайно, найважливішим є шлях до папки або файлу в операційній системі та саме ключове слово. Важливим уточненням до введення ключових слів є те що, введення має передбачувати що таких слів може бути декілька, тобто замість ключових слів також можуть бути використані комбінації ключових слів у словосполученні. Це також необхідно підкреслити, оскільки не завжди зрозуміло, чи введене слово встановлено як вхідні дані чи ні.

Висновки

Отже, всі прийняті рішення слід конкретизувати після детального вивчення окремих процесів і сформулювати загальні вимоги до системи.

Основними способами досягнення поставленої мети та їх напрямки у вирішенні проблеми будуть такі:

- Обробка даних у аудіофайлі за допомогою мовних моделей та навчання штучного інтелекту.
- Алгоритм Рабіна-Карпа – для точного пошуку за хеш-значенням

Були визначені параметри для аналізу контексту, а саме: розмір для дослідження контексту – 2 слова, а також напрямок – з лівого та з правого боків знайденого співпадіння, та розмір , але при цьому пріоритет віддавався наступним словам, якщо такі наявні, аналіз контексту може включати відношення до окремого речення чи абзацу, якщо така можливість буде при навченому штучному інтелекті і сформованих моделей.

Сформульовано вимоги до програмного продукту, які визначають функціональні вимоги до системи: тип файлу, призначений для обробки – аудіофайл формату WAV, оскільки він передає максимально можливу якість з файлу та в такий формат може бути конвертований будь який файл який має звукову доріжку.

Були виділені такі нефункціональні вимоги:

- Програмні – реалізація програмної системи за допомогою мов React Typescript та Kotlin Ktor
- Апаратні – мінімальними будуть процесори з частотою від 2 ГГц, новіше за 2018 рік, оперативна пам’ять від 8 ГБ, і вільне місце на жорсткому диску 100 МБ, також дослідження з приводу ефективності використання твердотілих накопичувачі, таким чином найшвидшим буде диск формату M2, потім SSD, та найповільнішим HDD. Та обов’язкова наявність звукової карти.
- Надійність та безпека – гарантована заборона телеметрії для додатку та відсутністю доступу до зміни вмісту та стану файлу, введені файли

матимуть властивість відкриття лише для читання на рівні операційної системи.

- Інтерфейсні – високі показники зручності та лаконічності інтерфейсу з малою кількістю вхідних даних, проте повинні мати ефективно заповнені звіти.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

3.1 Робота з мовними моделями

Для подальшого обговорення роботи з моделями для інтерпретації звуку варто визначити певні поняття.

Мовні моделі – це алгоритми чи статистичні моделі, що використовуються в обробці природної мови для розуміння та генерації тексту. Їхнє завдання полягає у прогнозуванні слів чи послідовностей слів у тексті на основі контексту. Вони використовують статистику, граматичні правила та інші методи для розуміння того, які слова чи фрази ймовірно слідують після певного текстового контексту. Мовні моделі є важливою складовою у багатьох областях обробки мовлення, включаючи розпізнавання мовлення, машинний переклад, генерацію тексту та інші завдання, де необхідно працювати з природною мовою.

Для створення та використання мовних моделей потрібно визначити мову моделювання. Мовою моделювання прийнято вважати якусь штучно створену мову, яка застосовується для вираження даних або знань за допомогою послідовного набору правил. Такі правила використовуються для інтерпретації значення структурних частин.

Текстові види інформаційних моделей можна виразити більш формалізованими природними мовами, а саме це є ніби словник з усіма можливими словами окремої мови. Попередня обробка тексту — це важливий етап у підготовці даних для подальшого аналізу чи використання в моделях машинного навчання. Цей словник також наповнюється різними відмінюваннями та варіаціями кожного слова. Для підготовки такого словника потрібно слідувати таким крокам:

- Токенізація – розбиття тексту на окремі слова або токени. Це може бути банальним розділенням тексту за пробілами, але також може включати обробку складних випадків, таких як розділення слів з апострофами, чисел та спеціальних символів.

- Перетворення у нижній регістр – стандартизація тексту до одного регістру. Це допомагає уникнути проблем зі словами, які можуть бути вказані по-різному, але насправді представляють одне й те ж слово.
- Вилучення стоп-слів – стоп-слова — це слова, які часто зустрічаються у тексті, такі як "і", "та", "у", "на", які не несуть суттєвої інформації для розуміння сутності тексту. Вони можуть бути вилучені для полегшення аналізу.
- Стемінг та лематизація – це процес перетворення слова до його базової форми. Стемінг скорочує слова до їх кореневих форм, тоді як лематизація перетворює їх до словникової форми. Наприклад, слово "бігають" після стемінгу може стати "біг", а після лематизації — "бігати".
- Вилучення непотрібних символів та чисел – видалення небуквених символів, чисел чи іншої непотрібної інформації, яка може заважати аналізу тексту.
- Усунення дублікатів та неправильних символів – виявлення та видалення дублікатів рядків або символів, що можуть виникнути при обробці даних.

Ці кроки бажано застосовувати в такій послідовності задля кращої точності моделі, але можливе використання у різному порядку, разом або окремо.

Граматика - це формальні правила, які описують прості правила побудови речень. Лексеми, отримані на етапі створення словника, намагаються зіставитися з граматикою і якщо вдало, то виводиться результат.

Потім створюються акустичні моделі для розпізнавання мовлення і визначення того, які акустичні характеристики відповідають різним звуковим одиницям, таким як фонеми або фонетичні одиниці. Їх створення відбувається в 2 етапи. Спочатку потрібно зібрати аудіодані з різних джерел. Зазвичай це включає запис мовлення людей у різних умовах та акустичних середовищах. Важливо збирати різноманітні дані, щоб модель була готова до різних ситуацій, таких як шуми, акценти, тембри голосу тощо. Чим більше різноманітних даних буде

доступно для навчання, тим краще модель може узагальнювати інформацію для подальшого розпізнавання мовлення. Після збору аудіоданих необхідно витягнути характеристики аудіосигналу, що представляють його акустичні властивості. Це може бути зроблено шляхом використання різних технік витягнення ознак, таких як MFCC (Mel Frequency Cepstral Coefficients), спектрограми, амплітудні характеристики, темп, енергія тощо. Ці ознаки допомагають в аналізі звуку та його характеристик, щоб підготувати дані для подальшого навчання моделі на базі цих характеристик.

Більш складним етапом моделювання програмної системи для розпізнавання мовлення є статистичні моделі. Статистичні моделі мови описують більш складну мову. Вони містять ймовірності слів і словосполучень. Ці ймовірності оцінюються на основі вибіркових даних і автоматично мають певну гнучкість. Можлива кожна комбінація зі словника, хоча ймовірність кожної комбінації буде різною. Наприклад, якщо ви створюєте статистичну модель мови зі списку слів, вона все одно дозволить декодувати словосполучення, навіть якщо це не було вашим наміром. Загалом сучасні інтерфейси розпізнавання мовлення мають тенденцію бути більш природними та уникати командно-контрольного стилю попереднього покоління. З цієї причини більшість розробників інтерфейсів віддають перевагу розпізнаванню природної мови за допомогою статистичної моделі мови.

Створені моделі записуються як текст, проте різні методи машинного навчання можуть передбачати їх компресію чи зберігання у іншому типі файлів. Розглянемо декілька з них. JSON або YAML – ці формати використовуються для збереження структурованої інформації у вигляді текстових файлів, вони можуть бути зручними для малих моделей або моделей з обмеженою складністю. Бінарні файли – це компактні файли, які забезпечують швидший доступ до інформації через їх бінарний формат, такі файли часто використовуються для зберігання складних моделей, наприклад, за допомогою бібліотеки Pickle у Python. Спеціалізовані формати – деякі бібліотеки та програми машинного навчання мають власні формати для збереження моделей, наприклад, формати TensorFlow (.pb, .h5), PyTorch (.pt), або Word2Vec (.bin). Бази даних - моделі можуть зберігатися в базах даних,

особливо якщо це частина великого корпусу текстів або інші великі об’єми даних.

3.2 Машинне навчання

Визначивши потрібні об’єкти для розпізнавання мовлення, потрібно вибрати інструмент для створення, використання та поповнення моделей. CMU Sphinx зараз є найбільшим проектом із розпізнавання людської мови. Sphinx — це система безперервного мовлення, незалежна від мовця, яка використовує приховані марковські акустичні моделі (HMM) і n-грамову статистичну модель мови. Він був розроблений Кайфу Лі, тайванським підприємцем у сфері розробки програмного забезпечення та штучного інтелекту. Sphinx продемонстрував можливість розпізнавання великого словникового запасу безперервної мови, незалежно від мовця, можливість якого в той час (1986) була предметом суперечок. Sphinx 4 — це повна переробка двигуна Sphinx з метою надання більш гнучкої основи для досліджень розпізнавання мовлення, повністю написаної на мові програмування Java. Sun Microsystems підтримала розробку Sphinx 4 і внесла в проект досвід розробки програмного забезпечення.

Рис. 3.2.1 Кайфу Лі – творень CMU Sphinx

У інструментарій входять такі програми та бібліотеки:

- Pocketsphinx – невелика програма, яка приймає на вхід довільні акустичні моделі, граматики та словники, а також звуковий потік, також це може бути звуковий файл або потік з мікрофона. На виході виходить розпізнаний текст. Написана на C.

- Sphinxbase — бібліотека, необхідна для роботи Pocketsphinx
- Sphinxtrain – програма для навчання акустичних моделей.

Для початку навчання потрібно обрати так званого в сфері штучного інтелекту – тренера, засіб який буде навчати штучний інтелект на основі внесених в нього даних. Вище було описано систему інструментарію для роботи штучного інтелекту у сфері розпізнавання мовлення. Програма Sphinxtrain розрахована саме для навчання найважчих моделей, а саме акустичних та статистичних, вона має відкритий вихідний код та активно підтримується розробниками.

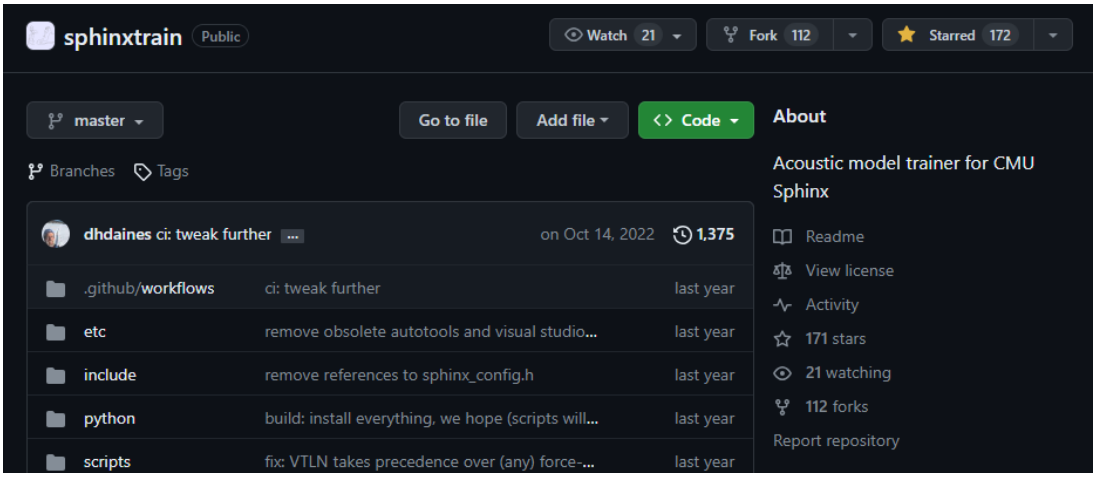

Рис. 3.2.2 Відкритий репозиторій Sphinxtrain на GitHub

Тренер вивчає параметри для моделей звукових одиниць за набором зразків мовних сигналів. Це називається навчальною базою даних. База даних містить інформацію, необхідну для отримання статистики з мовлення у формі акустичної моделі. Тренеру потрібно повідомити, які звукові одиниці ви хочете для вивчення, і принаймні послідовність, у якій вони відбуваються в кожному мовному сигналі у вашій навчальній базі даних. Ця інформація надається тренеру через файл, який називається файлом стенограми. Він містить послідовність слів і немовленнєвих звуків у тому самому порядку, в якому вони з’явилися в мовному сигналі, за якими йде тег, який можна використати, щоб зв’язати цю послідовність із відповідним мовним сигналом.

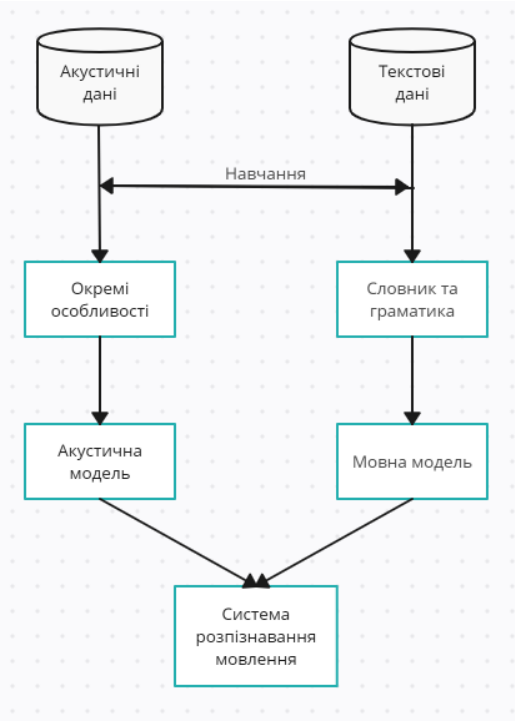

Рис 3.2.3 Діаграмне зображення взаємозв’язку моделей

Таким чином, на додаток до мовних сигналів і файлу транскрипції, тренеру потрібно отримати доступ до двох словників: один, у якому допустимі слова в мові зіставляються з послідовностями звукових одиниць, а інший у якому немовні звуки зіставляються з відповідними немовними або мовними звуковими одиницями. Перший будемо називати фонетичним словником мови. Останній називається словником-заповнювачем. Потім тренер переглядає словники, щоб отримати послідовність звукових одиниць, які пов’язані з кожним сигналом і транскрипцією. Після навчання обов’язково потрібно запустити декодер для перевірки результатів навчання. Декодер бере модель, перевіряє частину бази даних і еталонних транскрипцій і оцінює якість (WER) моделі. На етапі тестування використовується модель мови з описом можливого порядку слів у мові.

З документації Sphinx треба виокремити застереження та поради щодо збереження та формування моделей. База даних має добре відображати те, яку мову ви збираєтеся розпізнати. Наприклад, якщо метою розпізнавання є телефонні розмови, краще використовувати записи телефонних розмов. Якщо потрібно працювати з мобільним мовленням, краще знайти мобільні записи. Мова значно відрізняється в різних каналах запису. Мова, розшифрована з формату файлу mp3,

значно відрізняється від запису з мікрофона. Однак, якщо недостатньо записаного мовлення в необхідних умовах, обов’язково треба використати будь яке інше мовлення. База даних має містити записи достатньої кількості ораторів, різноманітні умови запису, достатню кількість акустичних варіацій і всі можливі мовні речення. Як згадувалося раніше, розмір бази даних залежить від складності завдання, яке ви хочете вирішити. База даних повинна мати дві частини, згадані вище: навчальну та тестову. Зазвичай тестова частина становить приблизно десяту від загального обсягу даних, але не рекомендовано мати більше 4 годин запису як тестових даних.

Після описаного алгоритму дій для навчання штучного інтелекту, в даному випадку для розпізнавання мовлення на окремі слова, та виокремлених застережень та порад від розробників інструменту для навчання потрібно описати як відбувався процес навчання штучного інтелекту для досліджуваної програмної системи. Обраною мовою для розпізнавання була англійська, як найбільш популярна міжнародна мова. Після встановлення програми Sphinxtrain було прийнято рішення шукати мінімально підготовлені моделі для подальшого навчання. Винятком були словник та граматика, ці моделі були у відкритому доступі з стандартним наповненням. Поповнення цих моделей слабко відноситься до навчання штучного інтелекту, воно більше стосується лінгвістичного дослідження. Хоча зміна цих моделей відбувалась у процесі навчання.

Для початку потрібно було створити базу даних за визначеним в документації скриптом та налаштувати конфігурацію для програми, надавши шляхи до кожної складової потрібних моделей. А також було підключено паралельне виконання на декількох ядрах для пришвидшення навчання. Для самого навчання потрібно було лише додати аудіофайли з записами у вказану в документації теку і запустити програму окремим скриптом.

```
Baum Welch starting for 2 Gaussian(s), iteration: 3 (1 of 1)
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Normalization for iteration: 3
Current Overall Likelihood Per Frame = 30.6558644286942
Convergence Ratio = 0.633864444461992
Baum Welch starting for 2 Gaussian(s), iteration: 4 (1 of 1)
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Normalization for iteration: 4
```

Рис. 3.2.4 Прогрес навчання моделі

Як було зазначено в документації, після завершення процесу навчання на файлах потрібно запустити декодер, який вирахує частоту помилки до слова (WER) та відобразить більш точний результат запущених процесів.

```
p   I   T      t   s   b   u   r   g   H      (MMXG-CEN5-MMXG-B)
p   R   EIGHTY t   s   b   u   r   g   EIGHT  (MMXG-CEN5-MMXG-B)
Words: 10 Correct: 7 Errors: 3 Percent correct = 70.00% Error = 30.00% Accuracy = 70.00
Insertions: 0 Deletions: 0 Substitutions: 3
october twenty four nineteen seventy  (MMXG-CEN8-MMXG-B)
october twenty four nineteen seventy  (MMXG-CEN8-MMXG-B)
Words: 5 Correct: 5 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00
Insertions: 0 Deletions: 0 Substitutions: 0
TOTAL Words: 773 Correct: 587 Errors: 234
TOTAL Percent correct = 75.94% Error = 30.27% Accuracy = 69.73%
TOTAL Insertions: 48 Deletions: 15 Substitutions: 171
```

Рис. 3.2.5 Результат виклику декодера

Під час виконання такого алгоритму стало цікаво, що ж саме програма для тренування робить, чому це так довго і як Sphinxtrain визначає що слово було інтерпретовано з помилкою. Дослідивши це питання я дійшов висновку що ця програма і є вже натренованим штучним інтелектом для тренування інших штучних інтелектів, по факту вона робить теж саме тільки з результатом максимально близьким до вірного та перевіряє співпадіння між моделями на яких працює штучний інтелект який тестується. І якби не було такої програми, то людина мала б сама співставляти правильність транскрипції та реальної інформації в аудіофайлі. Проте програма все рівно довго обробляє інформацію та сильно навантажує процесор комп’ютера.

Для навчання були обрані файли з відкритого зібрання архівів аудіювань, вони мали різні середовища запису, різних диктаторів та різні сфери для тренування. Під час дослідження також були спроби записати власні звукові файли для тренування штучного інтелекту, проте там відсоток помилок був більший. Можливо цьому

причина через відчутний акцент чи поганий звукозапис, але налаштована модель під ідеальне аудіювання погано розпізнавала таке мовлення.

В результаті навчання були сформовані моделі штучного інтелекту у визначених для них місцях, тепер створений контролер для штучного інтелекту зможе їх використовувати. Початкові розміри моделей не були записані перед навчанням, тому оцінка зросту моделі можуть мати неточності та отримані дані приблизні. Словник має окреме розширення .dict та займає 3,2 Мб, за власними спостереження змін, словник виріс близько 100-200 Кб, назва файлу cmudict-en-us. Мовна модель має розширення бінарного файлу та займає 26,5 Мб пам'яті, вона зросла менше ніж в половину. Також була автоматично створена мовна модель для мобільного запису з розміром в 838 Кб.

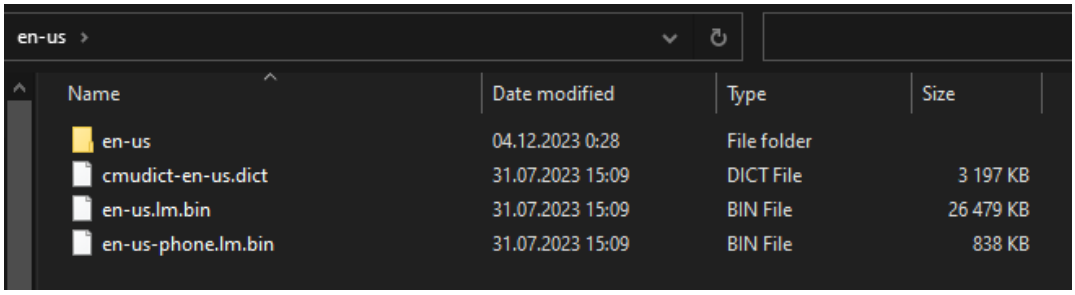

Рис.3.2.6 Файли словника та мовної моделі

Акустична модель є найбільшою, визначається текою з різними файлами, наповненість теки також може відрізнятись від спеціалізації штучного інтелекту який навчається. Акустична модель займає 52,3 Мб та зростала вона приблизно з 2-5 Мб, тобто зріст був найбільший, що було очікувано. Найбільшими файлами з акустичної моделі є файл значень – 23 Мб та файл варіацій – 23 Мб.

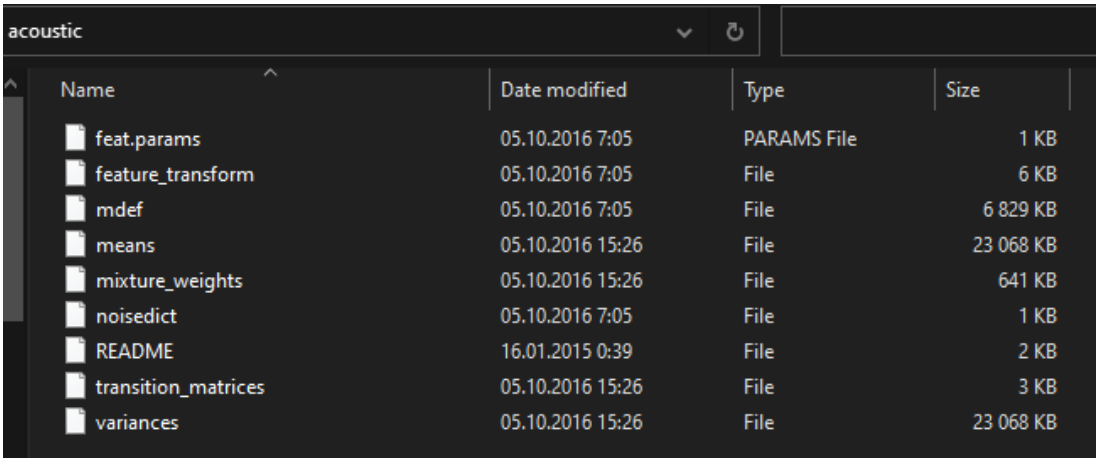

Рис.3.2.7 Файли акустичної моделі

У результаті навчання моделей для штучного інтелекту, моделі займають 82 Мб та мають відсоток правильно результату 80,8 відсотків. Такі показники не є найкращими, проте це дослідження проводилось однією людиною за період близько місяця, тому це непоганий результат.

3.3 Імплементування штучного інтелекту в програмну систему

Оскільки навчання моделей для штучного інтелекту відбувалось за допомогою програми Sphinxtrain, то коли постало питання пошуку бібліотеки для використання штучного інтелекту вибір пав на Sphinx4. Це бібліотеки від CMU Sphinx, тих самих лідерів у розпізнаванні мовлення в аудіофайлах. Ця бібліотека написана на мові Java, що є надзвичайно підходящим варіантом, оскільки розроблена програмна система написана на Kotlin і ця мова має повну підтримку коду написаного на Java.

Для підтримки бібліотеки був написаний окремий клас SpeechToTextRecognizer, який мав створювати основний клас з бібліотеки для розпізнавання мови StreamSpeechRecognizer зі створенням конфігурації для роботи бібліотеки. Там потрібно вказати основні дані, а саме шлях до словника, акустичної та мовної моделі, а також можна конфігурувати власні налаштування для штучного інтелекту.

```
private fun createSpeechRecognizer(): StreamSpeechRecognizer {
    val configuration = Configuration()

    configuration.acousticModelPath = "file:///C:/Users/Nick/Desktop/acoustic"
    configuration.dictionaryPath = "file:///C:/Users/Nick/Desktop/en-us/cmudict-en-us.dict"

    configuration.languageModelPath = "file:///C:/Users/Nick/Desktop/en-us/en-us.lm.bin"

    configuration.sampleRate = 44100
   // configuration.setFloat("-inputChannels", 1);

    configuration.grammarPath = "src/main/resources"
    configuration.grammarName = "text"
    configuration.useGrammar = true

    return StreamSpeechRecognizer(configuration)
}
```

Рис. 3.3.1 Конфігурація для штучного інтелекту

І після створення конфігурації вже можна було використовувати цю бібліотеку. Для реалізації роботи розпізнавання мовлення та перетворення аудіофайлу в текст було написано ще 2 методи цього класу.

```
new *
fun recognizeSpeech(audioFile: File): String {
    val stream = audioFile.inputStream().buffered()
    stream.skip(44)
    return recognizeSpeech(stream)
}

new *
private fun recognizeSpeech(inputStream: BufferedInputStream): String {
    recognizer.startRecognition(inputStream)
    var resultText = ""
    var result: SpeechResult? = recognizer.result
    while (result != null) {
        resultText += result.hypothesis + " "
        result = recognizer.result
    }
    recognizer.stopRecognition()

    // for logging
    // println(resultText)
    return resultText
}
```

Рис. 3.3.2 Методи обробки результатів

Перший метод має відкривати файл як потік даних вказаного файлу та виконувати дуже важливу дію, а саме пропуск перших 44 бітів файлу, бо це той об’єм файлу де знаходяться непотрібні бібліотеці метадані файлу. Це було описано в документації до бібліотеки Sphinx4 та наголошено про використання окремого методу. А другий має дочекатись повної обробки файлу та повернути результат.

3.4 Огляд та опис використаних алгоритмів для пошуку

Наступним етапом розробки було імплементування функціоналу пошуку і одразу стало зрозуміло що така робота з штучними інтелектом була не ефективною, оскільки паралельно може тривати пошук, доки файл проходить процес розпізнавання в ньому мови. Для цього потрібно було розпаралелити процеси, найкраще це можна зробити за допомогою багатопоточності – це концепція, яка використовується для одночасного виконання кількох завдань або процесів в програмі або системі, кількість доступних потоків залежить від кількості ядер

процесора. Але Kotlin має дуже зручну та корисну особливість – корутини – це легковажні обчислювальні відомості, які дозволяють вам виконувати асинхронні операції без блокування потоків. Вони працюють у межах одного потока, але можуть виконувати зупинку та відновлення безпосередньо всередині функції, що дозволяє здійснювати асинхронні операції.

Алгоритм був перероблений так – окрема корутина займається розпізнаванням мови за допомогою бібліотеки та моніторить наявність результату від процесу, доки інша корутина перевіряє кожне слово з результату на співпадіння з ключовим словом. Процес розпізнавання мови динамічно, як тільки має розпізнане слово повертає його у вигляді результату. А процес перевірки на ключове слово паралельно зберігає та оновлює слова зліва від ключового відповідного розміру контексту, при співпадінні з шуканим словом активується інший алгоритм програми – збереження результату, тобто записується в результат співпадіння, контекст до ключового слова, порядковий номер слова, часовий проміжок в якому зустрілось це слово, а також цей результат чекає доки будуть розпізнані наступні слова контексту за шуканим словом, вони також додаються до результату і пошук продовжується.

З переробленим алгоритмом змінилась думка і щодо використання алгоритмів пошуку. Оскільки штучний інтелект для розпізнавання мовлення працює повільніше ніж звичайне співставлення слів, то можна відразу відкинути такі алгоритми як Бойера-Мура та Кнута-Моріса-Пратта, їхні особливості щодо прискорення пошуку в тексті нівелюються. Була відтворена спроба використання алгоритму Рабіна-Карпа, використання хеш-значень для порівняння ключового слова та частини тексту для перевірки, проте застосування цього алгоритму не показало покращення ані швидкості, ані якості пошуку. Це ще раз доказує те що весь процес значно сповільнює розпізнавання мовлення штучним інтелектом.

Також були застосовані алгоритми статистичного аналізу. Для цього потрібно було налаштувати програму тренер для штучного інтелекту Sphinxtrain, щоб вона помічала певні повторювані шаблони використання слів. Ці дані записувались разом і в мовну, і в акустичну моделі. Статистичний аналіз використовує дані про певну закономірність порядку та комбінації слів задля прогнозування наступного слова чи

цілих частин речення. Використання статистичного аналізу пришвидшило розпізнавання мовлення, де використовуються стандартні фрази чи словосполучення, також було помічено що штучний інтелект на всякий випадок перевіряє на точне співпадіння з прогнозуванням, але це справді пришвидшує процес. Коли різні часті фрази використовують якесь спільне слово, то ці фрази перевіряються разом і точність розпізнавання підвищується. З цього також можна зробити висновок що якість розпізнавання напряму залежить від кількості та якості даних що має штучний інтелект.

3.5 Реалізація знаходження контексту

Аналіз контексту є ключовим етапом обробки аудіофайлів, що має просту реалізацію у програмуванні. У плануванні системи було визначено, що оптимальний розмір контексту – 2 слова. Хоча важливий акцент робиться на наступних словах, ми вирішили використовувати обидва напрямки для точності. Розділові знаки стануть частиною слів, а контекст може включати сусідні граматичні конструкції, такі як речення, але не абзаци.

Здавалося б кілька рядків коду мають вирішувати таке завдання, навіть якщо потребують зберігання двох слів перед знайденим, що вимагає місця в пам'яті. Але оскільки отримання результату розпізнавання мовлення динамічне, тобто як тільки було розпізнане слово – воно буде повернуто, то потрібно мало того що зберігати попередні слова, а ще й чекати наступні 2 слова, що призвело до розділення реалізацій моніторингу результатів обробки штучного інтелекту. Важливим випадком залишається коли одне ключове слово іде одразу після іншого ключового слова. Для цього був розроблений механізм вивантаження даних після співпадіння, щоб не змінювати стан пошуку. Таким чином пошук є повністю незалежним та завершиться лише після повної обробки файлу. Пошук зберігає всі необхідні дані та ніхто на них не впливає, а саме попередні 2 слова контексту та слово що перевіряється. Після співпадіння ключового слова пошуку з словом у тексті відбувається вивантаження, викликається метод який формує частину звіту, в нього

передаються копії 2 попередніх слів контексту, ключове слово, порядковий номер слова та часовий проміжок де відбулось співпадіння, а також цей метод чекає розпізнавання наступних 2 слів контексту, коли вони будуть знайдені то запишуться в цю частину звіту та відразу будуть відправлені для форматування загального звіту.

Отже, алгоритм дослідження контексту дещо ускладнився з імплементуванням розпізнавання мовлення штучним інтелектом через динамічне отримання результатів, але ця проблема була вирішена оптимізованим в плані використання оперативної пам’яті та потужності процесора.

3.6 Реалізація користувацького інтерфейсу

Для реалізації користувацького інтерфейсу була обрана комбінація технологій React та TypeScript, що зробить якість фронтенду вище, користувацький інтерфейс зручніше, а розробку простіше.

React – це JavaScript-бібліотека для розробки користувацького інтерфейсу. Вона створена компанією Facebook та має широке застосування веб-розробки для створення інтерактивних і швидких веб-сайтів та веб-додатків.

Основні особливості React:

- Компонентний Підхід – React базується на компонентній архітектурі. Це означає, що весь інтерфейс може бути розбитий на невеликі незалежні компоненти, які можна легко керувати, оновлювати та повторно використовувати.
- Virtual DOM – у React використовується віртуальний DOM (Document Object Model), що дозволяє йому ефективно взаємодіяти з реальним DOM. Він створює віртуальне представлення сторінки у пам'яті, здатне оперативно оновлюватись, що робить маніпуляції з DOM швидшими та ефективнішими.
- Односторінкові Додатки (SPA) – React дозволяє створювати односторінкові додатки, які не потребують перезавантаження сторінки під час навігації. Це забезпечує плавний та швидкий досвід для

користувача.

- Стан та властивості – у React використовуються стан (state) та властивості (props) для керування даними і передачі їх між компонентами. Стан - це внутрішні дані компоненту, а властивості - це дані, які передаються вниз від батьківського компонента до дочірніх.
- Розширені Можливості – є велика кількість додаткових бібліотек і інструментів для розширення можливостей React, таких як Redux для управління станом додатка, React Router для маршрутизації, Material-UI або Bootstrap для створення графічного інтерфейсу тощо.

React став популярним завдяки своїй простоті, продуктивності та активній спільноті розробників. Він дозволяє ефективно створювати високоякісні користувацькі інтерфейси для різноманітних веб-додатків.

А також була застосована мова TypeScript – це мова програмування, розроблена компанією Microsoft, яка є розширенням JavaScript з можливістю статичного типізування. Основна ідея TypeScript полягає в тому, щоб додати до JavaScript ряд додаткових функцій, що дозволяють розробникам працювати з більш складними проектами, покращуючи безпеку та підвищуючи продуктивність. Ця мова дозволяє вказувати типи даних для змінних, параметрів функцій, об'єктів та інших об'єктів, допомагає виявляти помилки під час розробки, забезпечуючи більшу надійність коду на відміну від JavaScript. Проте має повну сумісність з останньою, оскільки є її розширенням. Таким чином немає ніяких проблем комбінувати React та TypeScript.

У розробці програмної системи використовувався компонентний та функціональний підхід, де об’єктом побудови архітектури веб-клієнту складає компонент, а оперативними можливостями є функції. Другим способом було створення класів, як у всіх об’єктно орієнтованих мовах, проте використання класів на фронтенді є дещо складнішим, та доцільність цього досі лишається спірним питанням.

Компоненти можуть бути різної вкладеності та складності, деякі з них можуть мати всередині один компонент, проте мати багато функцій обробників, які

передаються в дочірній компонент як властивості. Такими в програмній системі виступають Layout та ProcessSettingsForm. Також React має окрему функціональність названу хуками, це такі методи які мають складну логіку, можуть бути використані в будь-якому компоненті але можуть лише віддавати вихідні дані чи функції які будуть незалежно оброблювати вхідні дані. Я розробив один хук для власних потреб useInterval, який з проміжком вказаного часу має робити вказану дію.

Рис.3.6.1 Стандартний хук в React

Також компоненти сильної вкладеності, такими компонентами є будь-які компоненти представлення, типу налаштувань, звітів, деталей процесу, завершений процес та інші. Такі компоненти просто складаються з правильного набору та послідовності компонентів поменше.

```
return (
    <Dialog
        open={open}
        onClose={handleClose}
        scroll={"body"}
        aria-labelledby="scroll-dialog-title"
        aria-describedby="scroll-dialog-description"
        fullWidth={false}
        maxWidth={"xl"}
        sx={{marginTop: '2%'}}
    >
        <DialogTitle sx={{background: '#5f0a87'}}/>
        <DialogContent sx={{background: 'linear-gradient(326deg, #a4508b 0%, #5f0a87 74%)'}}>
            <List sx={{width: '100%'}}>
                {props.info.map((file : FileInfo , index : number ) => {
                    return (
                        <ListItem key={index} sx={{border: "3px solid rgba(249, 105, 14, 1)", marginTop: '2%', borderRadius: 5}}>
                            <FileDetails file={file}/>
                        </ListItem>
                    )
                })}
            </List>
        </DialogContent>
        <DialogActions sx={{background: "#a4508b"}}>
            <GeneralButton onClick={handleClose}>Cancel</GeneralButton>
            <MainButton onClick={handleClose}>Print</MainButton>
        </DialogActions>
    </Dialog>
)
```

Рис.3.6.2 Приклад складної вкладеності компонента

І останнім типом компонентів є стильові компоненти – такими називаються ті що мають часте використання та при цьому однаковий набор стилю, тоді можна їх винести в окремі стильові компоненти для перевикористання.

```
5+ usages   NickKhilko *
export const Block = (props: BoxProps) => {
    return <Box sx={{
        display: 'flex',
        flexWrap: 'wrap',
        width: '100%',
        justifyContent: 'center',
        alignItems: 'center',
        marginTop: '1%'
    }} {...props}/>
}
```

Рис.3.6.3 Приклад стильового компоненту

Зібрання усіх потрібних компонентів в правильній послідовності з лаконічним використанням стилів представляє увесь клієнтський інтерфейс, а зі зв’язком до серверної частини становить повністю функціональну програмну систему для аналізу вмісту аудіофайлів. Детальний огляд та опис загального вигляду користувацького інтерфейсу буде розглянуто в 4 розділі під час тестування програми

3.7 Діаграмне зображення логіки опрацювання тексту

Для більшої зрозумілості інформації зазвичай використовують візуальні моделі, як от, наприклад, діаграми. Стандарт UML є широко використовуваною мовою графічного представлення та характеризації об’єктів для моделювання бізнес-процесів, прототипів та організаційної структури проектів у сфері інформаційних технологій.

Діаграма розгортання – це один з ключових інструментів для відображення середовища, де виконується програма. Вона не лише показує фізичні складові, необхідні для роботи, але і надає специфікацію цих елементів. На рівні програмного забезпечення вона включає бібліотеки та їх версії, операційні системи та сумісні браузери, а також індивідуальні бібліотеки, що підтримують цю систему.

На рівні апаратного забезпечення вона описує характеристики комп'ютера та необхідні ресурси, а також структури. Діаграма містить розділення на сховища, такі як хмарні та фізичні. У конкретній програмній системі є два сховища, хоча сама програма не використовує хмарні технології. Першим є фізичне – використовується серверною частиною програми для збереження історії процесів, але це локальне сховище іншим чином не використовується.

Друге - сховище браузера - незважаючи на його вид, воно функціонує окремо від першого, використовуючи інтернет-браузер для зберігання даних про прогрес запущених процесів, спрямованих на оптимізацію відображення в графічному інтерфейсі.

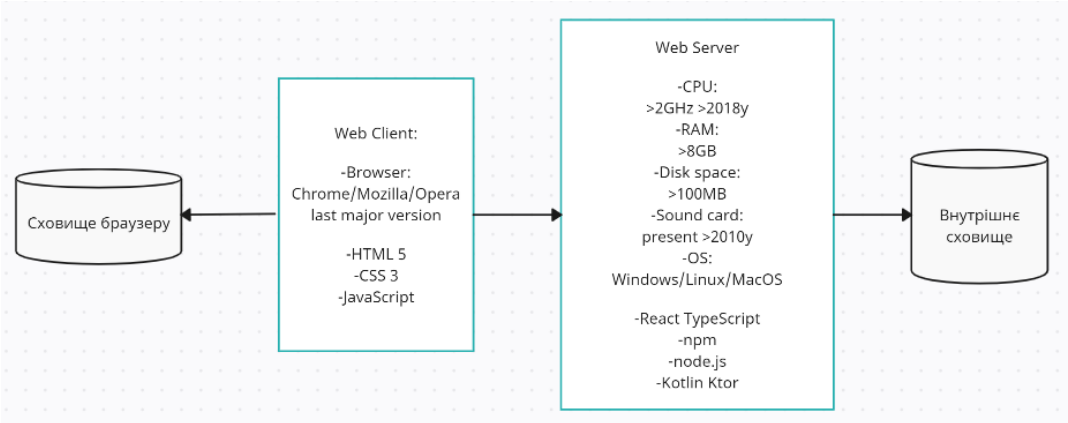

Рис. 3.7.1 Діаграма розгортання

Ще однією з основних діаграм, що використовується для відображення зв’язків у програмних продуктах, є діаграма класів. Діаграма класів — це вид діаграми, що використовується в об'єктно-орієнтованому програмуванні для візуального відображення класів системи, їх атрибутів, методів та зв'язків між ними. На такій діаграмі кожен клас зображується у вигляді прямокутника з вказанням назви класу. Зв'язки між класами показуються за допомогою ліній, які показують взаємозв'язки та взаємодію між класами. У ній перераховуються всі створені класи та інтерфейси з їх полями та методами, а також вказуються типи, параметри та типи результатів виконання. Крім того, зв’язки між класами можуть мати різні типи, такі як асоціація, агрегація та композиція. Крім цього, взаємозв’язки характеризуються за допомогою кількісних відношень, які можуть бути:

- Один до одного 1..1
- Один до багатьох 1..*, або навпаки *..1
- Багато до багатьох ..

Ця діаграма дозволяє відображати структуру системи, її складові елементи, та залежності між класами, що спрощує розуміння взаємодії компонентів програми та розробку коду.

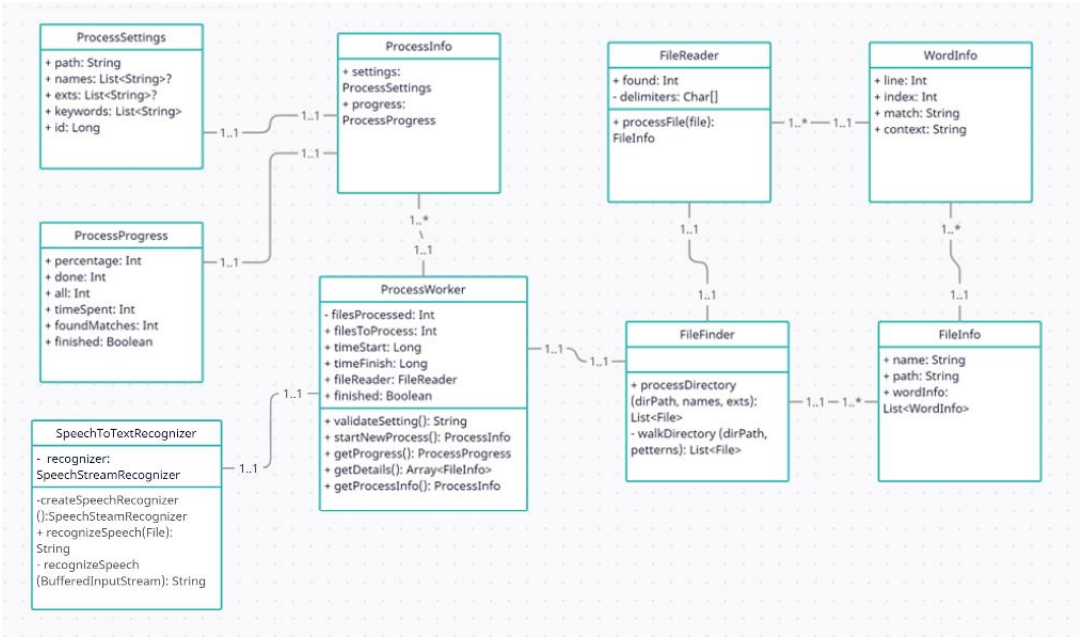

Рис. 3.7.2 Діаграма класів програмної системи.

Висновки

У висновку до цього розділу можна підсумувати успішним виконанням мети дипломної роботи, а саме розробка програмної системи для аналізу вмісту аудіофайлів, оцінка та огляд результату її роботи відбудеться в наступному розділі.

Було досліджено, створено та використано моделі для навчання штучного інтелекту, було проведено аналіз роботи та імплементування штучного інтелекту до системи. Етапи навчання, слідування документації розробників та проблеми у процесі навчання було розглянуто та представлено результати клопіткої роботи з програмою Sphinxtrain. Було презентовано мовну, акустичну та статистичну моделі для штучного інтелекту.

Для серверної частини обрано мову програмування Kotlin та фреймворк Ktor. На цьому етапі роботи було успішно відтворено алгоритм пошуку слів, а саме Рабіна-Карпа з основною думкою хешування шуканого слова та тексту в якому воно аналізується, описано алгоритм пошуку контексту та роботу нового алгоритму згідно з використанням даних розпізнавання мовлення штучним інтелектом.

Для клієнтської частини проекту було обрано фреймворк React та мову TypeScript та описано їхні переваги між іншими опціями для використання у фронтенд розробці. Було представлено основні компоненти та розділено на певні частини що можна об’єднати навколо мети чи схожих характеристик.

Було представлено зображення діаграм що описують архітектуру серверної частини системи та потрібну апаратну архітектуру для використання програмної системи

РОЗДІЛ 4
ХАРАКТЕРИСТИКА ОСНОВНИХ ПІДХОДІВ ДО АНАЛІЗУ ВМІСТУ АУДІОФАЙЛІВ

У процесі формування вимог виділено важливу мету: оптимізація користувацького інтерфейсу. Дослідження схожих програм показало, що інтерфейс може бути надто важким для користувача. Зазвичай, для них ключовою є можливість швидко розпочати роботу з налаштуваннями за замовчуванням, необов'язкові фільтри знаходяться у другому плані. Тому було акцентовано на створенні зручного, сучасного дизайну.

4.1 Огляд клієнтського інтерфейсу

Перед проходженням до наступної частини, важливо розглянути значення графічного інтерфейсу. Графічний інтерфейс є не тільки важливою складовою програмної системи, але і має великий вплив на:

- Привертання нових користувачів – якісний дизайн інтерфейсу значно впливає на перший враження користувача, що може сприяти створенню позитивного враження про бренд. Ефективний дизайн може значно відрізняти ваш продукт та спонукати людей до його використання.
- Збереження клієнтів – зручний інтерфейс, який спрощує навігацію та надає необхідну інформацію, заохочує користувачів використовувати систему. Це зменшує кількість відмов і збільшує відсоток активних користувачів, що є основною метою сучасних дизайнерів. Позитивне сприйняття вашого сайту також сприяє більшій лояльності до бренду, що є ключовою стратегією утримання клієнтів.
- Мінімізація витрат – якщо інтерфейс стає зрозумілим для користувача, це не лише зменшує кількість відмов, але також мінімізує кількість скарг, що надходять до служби підтримки користувачів. Грамотно спроектований дизайн також дозволяє виправляти помилки в навігації перед тим, як вони стануть причиною втрат – часу, коштів або активних

користувачів. Це дозволяє видаляти функції та можливості, які застаріли або неактуальні, що сприяє створенню більш функціонального дизайну програмного продукту.

- Задоволеність користувачів – задоволеність користувачів сприяє їхній лояльності до продукту. Швидке та ефективне задоволення потреб користувачів підвищує їхню лояльність, що в свою чергу збільшує обсяги продажів та мінімізує витрати ресурсів у всіх напрямках.

Графічний інтерфейс максимально залежить від користувацького досвіду (UX). Користувацький досвід - це спосіб, яким люди взаємодіють із програмними продуктами або сервісами. Це включає в себе враження, яке користувач отримує під час використання, його емоції, зручність, задоволення від взаємодії та відповідність його очікуванням. Важливість користувацького досвіду полягає у тому, що він напряму впливає на те, наскільки ефективно та задовільно користувач може використовувати програму чи сервіс. Гарний користувацький досвід сприяє збільшенню відвідуваності, залученню користувачів, підвищенню лояльності та задоволеності клієнтів. Коли продукт або сервіс легко зрозуміти та приємно використовувати, це сприяє позитивній репутації, збільшенню конкурентоспроможності та може впливати на прибутковість бізнесу. Основна ідея полягає в тому, щоб розуміти потреби та очікування користувачів та створювати продукти, які будуть простими у використанні, функціональними та тими що приносять задоволення від взаємодії. Глибоке розуміння користувацького досвіду дозволяє розробникам покращити продукт, зробити його більш ефективним та відповідним потребам користувачів.

З цією базою у нас є змога детальніше розглянути основні елементи інтерфейсу. Графічний дизайн представлений у зелено-блакитній кольоровій гамі. Перший використовується для фону, а другий – для обрамлення. Колір тексту білий, що забезпечує кращий контраст з фоновим кольором. Це можна віднести до темної тематики, але зберегти при цьому приємність сприйняття. Користувача вітає лише одна кнопка, яка активізує перший етап простого інтерфейсу. Цей принцип спрощує взаємодію до простої дії з натиском лише однієї кнопки. Надалі відкривається

форма з чотирма полями для тексту, з яких лише два обов'язкові: шлях до теки або файлу та поле для введення ключових слів. Також є дві кнопки: запуск процесу та закриття форми.

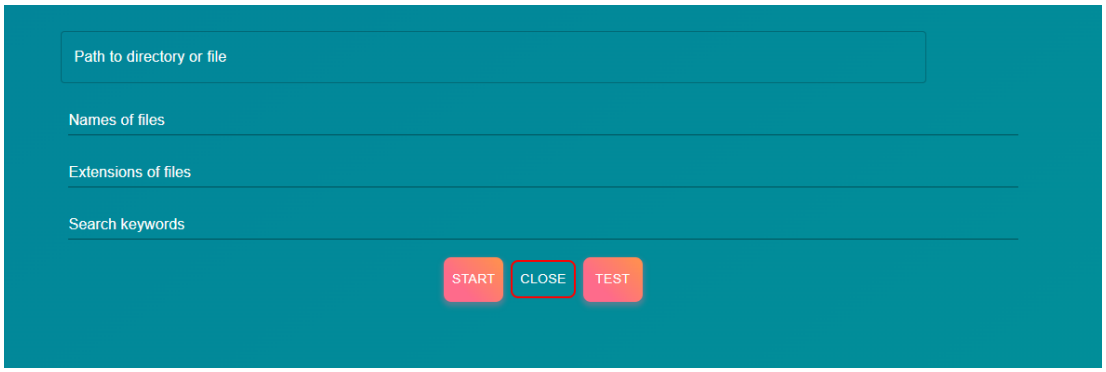

Рис. 4.1.1 Форма для налаштувань

Після натискання клавіші "Enter" введені дані виділяються рамками, що позначає їх фіксацію. Цю фіксацію можна скасувати, натиснувши на хрестик або просто стерши текст у полі. Текст, який вказує призначення поля, зміщується вгору, надаючи більший пріоритет введеним даним користувача, але зберігаючи видимість інформації, яка може вже не потрібна. Переглянемо Рис.4.1.2

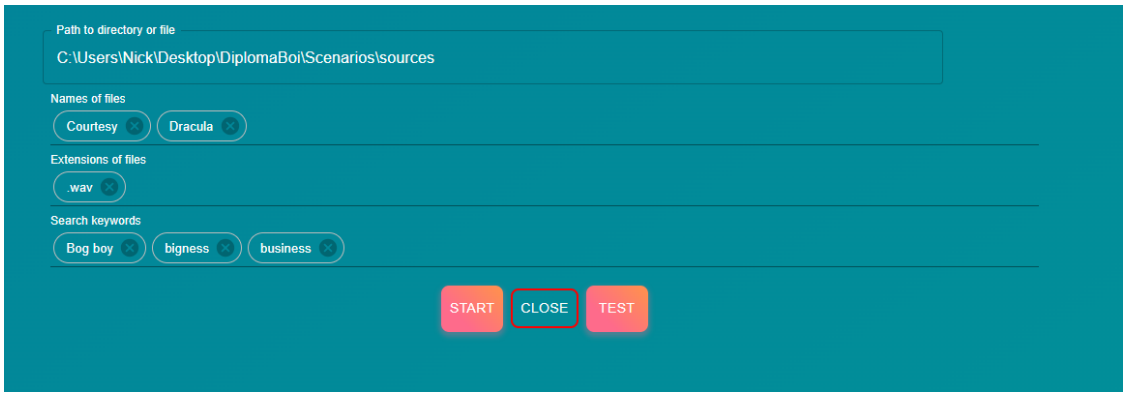

Рис. 4.1.2 Заповнена форма

Після активації процесу в браузері серверна частина негайно починає обробку з оновленням прогресу через певний інтервал часу. Це оновлення включає інформацію про опрацьовані файли, кількість знайдених файлів, витрачений час, співпадіння з ключовими словами та загальний прогрес у відсотках, що відображається графічно. Прогрес в інтерфейсі Рис. 4.1.3.

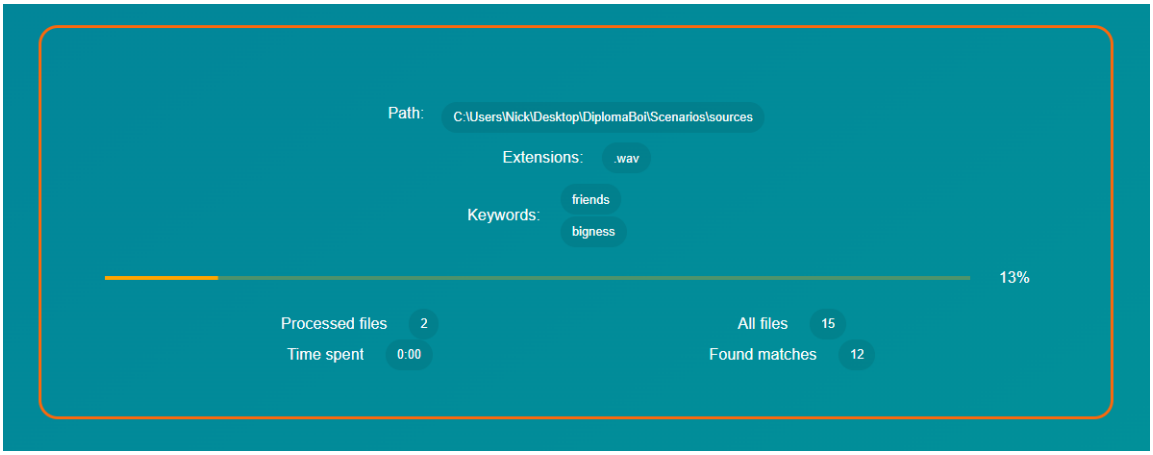

Рис. 4.1.3 Прогрес

Коли процес завершується, користувач отримує підтвердження у віконці знизу ліворуч (Рис. 4.1.4). Якщо користувач не приділяв увагу екрану, це може привернути його увагу до завершення роботи. У випадку, якщо користувач використовує іншу програму, активує повідомлення в браузері для нагадування про завершення роботи.

Рис. 4.1.4 Повідомлення про завершення пошуку

Після завершення процесу (Рис. 4.1.5), на додаток до сповіщення, список історії завершених процесів оновлюється, включаючи узагальнену інформацію про прогрес та заповнені форми. Це забезпечує зручний перегляд минулих процесів, який залишається незмінним при перезапуску додатку, проте може бути видалений, якщо він вже неактуальний.

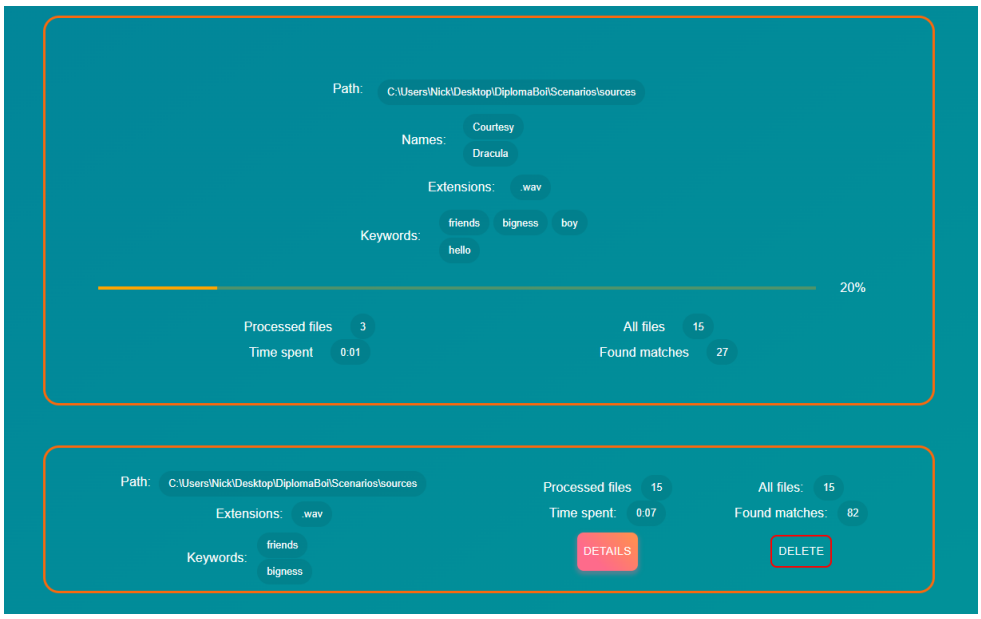

Рис. 4.1.5 Історія виконаних процесів

Основні результати роботи програми відображаються в окремому вікні після натискання кнопки "Деталі" (Рис. 4.1.6). Тут представлена вся інформація про результати знайдених співпадінь. Файли розділяються на частини, позначаючи шлях та назву файлу вгорі, а нижче – знайдені співпадіння з ключовими словами, а також контекст у квадратних дужках. Незалежно від кількості співпадінь, звіт містить всі файли для повного розуміння. Також передбачена можливість збереження звіту до файлу та його друку.

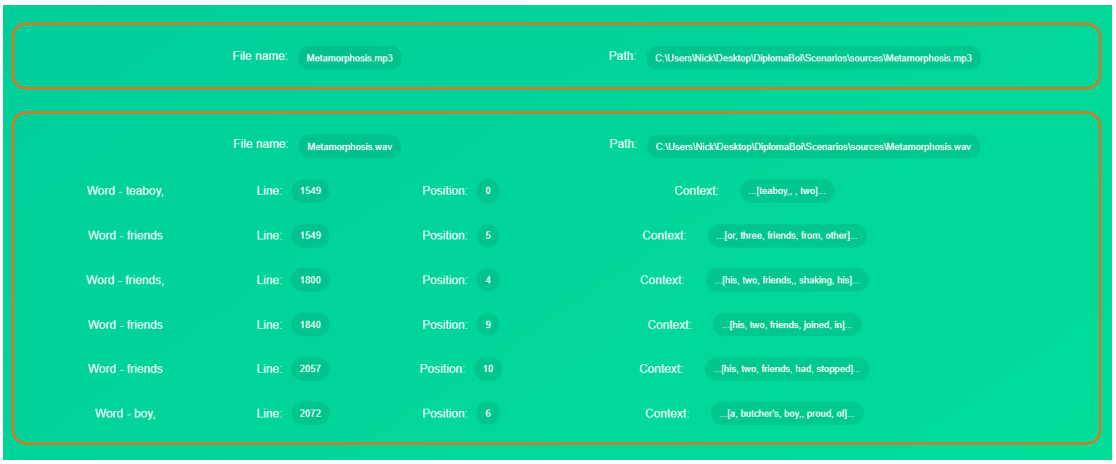

Рис. 4.1.6 Деталі виконання

Мінімальна кількість фільтрів, лаконічність і інформативність звітів роблять додаток максимально ефективним і простим у використанні. Палітру кольорів можна змінити, щоб зробити її більш офіційною, адаптуючи до вподобань користувачів, це не перешкоджає використанню програмного продукту. Сучасний

інтерфейс та доступність програми дозволяють новим користувачам легко розпочати роботу, а досвідченим – швидко опанувати рутину. Діаграма компонентів (Рис. 4.1.7) клієнтської частини системи також підтверджує простоту і ефективність графічного інтерфейсу.

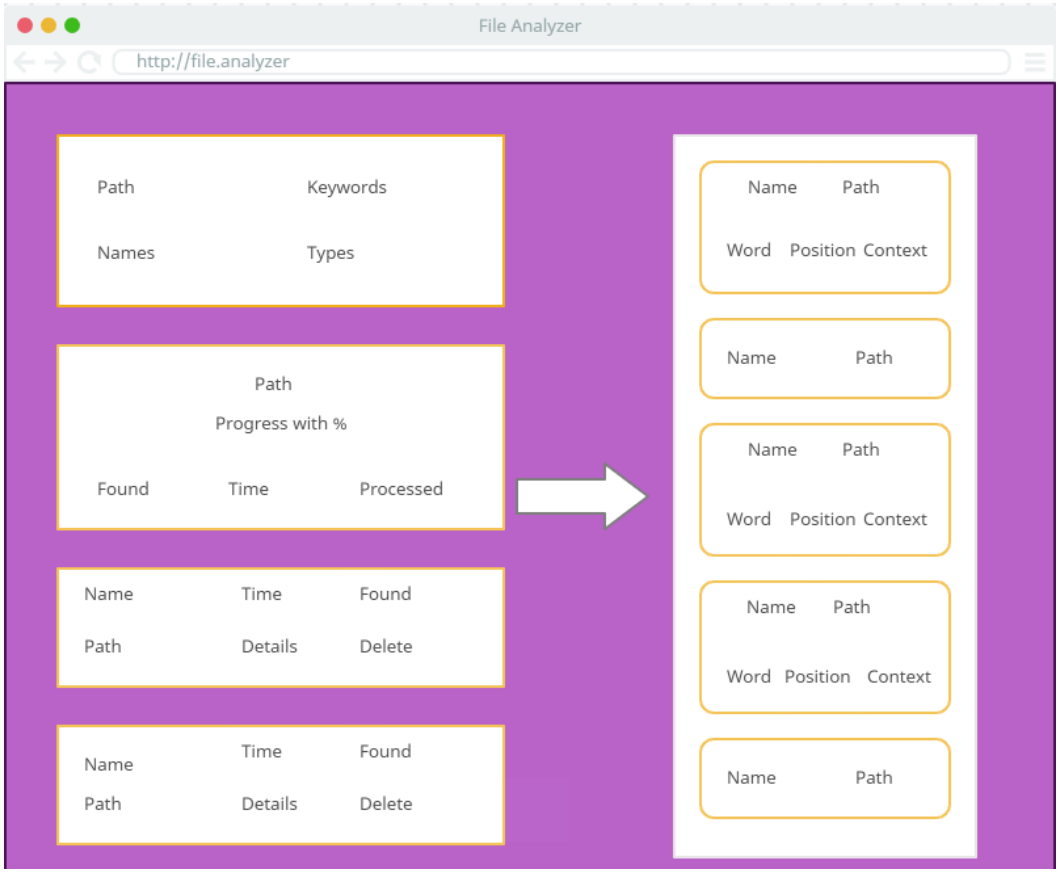

Рис. 4.1.7 Діаграма компонентів

4.2 Оцінка зручності інтерфейсу

Для перевірки зручності запропонованого інтерфейсу, корисно розглянути функціональні точки впливу на дані та транзакції. Аналіз функціональних точок є стандартним методом для вимірювання обсягу програмної системи з точки зору користувацького досвіду. Цей підхід незалежний від технологічної платформи, що використовується для розробки програми, і може застосовуватися для будь-якого продукту. У цьому методі ключовими є функціональні точки - складові системи, які визначають основні функції програми та оцінюють їх складність.

Підрахунок функціональних точок для даних починається з визначення

критеріїв оцінки:

- DET - унікальні поля даних.
- RET - логічна група даних, складена з унікальних полів.
- ILF - внутрішній логічний файл.
- EIF - зовнішній інтерфейсний файл.

Після цього проводиться групування даних та розподіл отриманих і відправлених форм даних на окремі компоненти. Перша група складається з полів для пошуку ключових слів, шляху до теки чи файлу та фільтрів для файлів: назва та тип. Ця група має 1 RET та 4 DET. Друга група містить поля, що отримуються під час процесу та в загальному висновку, такі як назва файлу, шлях, час та знайдені співпадіння. Ця група також має 1 RET та 4 DET. Нарешті, третя група включає інформацію про знайдені співпадіння: слово, номер лінії, індекс в лінії та контекст. Ця група також має 1 RET та 4 DET. В сумі отримуємо 3 RET та 12 DET, тобто по 4 поля на кожну логічну групу.

Тепер давайте розглянемо кількість файлів. У нас є лише 1 внутрішній логічний файл, який містить історію процесів та їх дані, і 1 зовнішній інтерфейсний файл - це зовнішнє сховище в браузері для запам’ятовування та оновлення прогресу запущеного процесу. Ці налаштування дозволяють оновлювати сторінку, не втрачаючи прогрес на сервері. Отже, у нас є 1 EIF (зовнішній інтерфейсний файл) та 1 ILF (внутрішній логічний файл). Тепер давайте оглянемо та оцінимо їх відповідно до таблиць 4.1 та 4.2.

Таблиця 4.1

Складність у співвідношенні DET до RET

	1-19 DET	**20-50 DET**	**50+ DET**
1 RET	Низьке	Низьке	Середнє
2-5 RET	Низьке	Середнє	Високе
6+ RET	Середнє	Високе	Високе

Таблиця 4.2

Складність для кількості ILF та EIF

Складність збереження даних	Кількість UFP (ILF)	Кількість UFP (EIF)
Низька	7	5
Середня	10	7
Висока	15	10

З обох таблиць випливає, що дані мають низький рівень складності збереження.

Тепер перейдемо до розрахунку функціональних точок для транзакцій. Транзакції - це цілісні процеси, що не можуть бути розділені. Це також функції, які змінюють стан програми. В цьому методі є різні типи транзакцій:

- EI - зовнішня вхідна транзакція, проста
- EO - зовнішня вихідна транзакція, що генерує дані на інтерфейсі
- EQ - зовнішній запит, складна транзакція
- FTR - кількість задіяних внутрішніх чи зовнішніх логічних файлів
- DET - унікальне поле даних

Тепер давайте обчислимо кількість EI, які є найпростішими операціями, наприклад, кнопка, яка не викликає складних обчислень. Таким чином, кнопка видалення процесу - 1 EI. Зовнішніх вихідних транзакцій у нас є: створення форми, відображення прогресу, виведення результату процесу та деталі процесу - 4 EO. Пора підрахувати кількість зовнішніх запитів на складні транзакції: надсилання результатів форми (запуск процесу), оновлення даних про прогрес та запит деталей процесу - 3 EQ.

Тепер час підсумувати обчислення кількості функціональних точок для різних типів транзакцій згідно з таблицею 4.3:

Таблиця 4.3

Складність транзакцій до їх кількості

Складність транзакції	Кількість UFP (EI та EQ)	Кількість UFP (EO)
Низька	3	4
Середня	4	5
Висока	6	7

Таким чином, щодо кількості зовнішніх вхідних транзакцій, оцінка складності визначається як низька, у той час як кількість зовнішніх вихідних транзакцій та запитів оцінюється як середня.

Оцінимо складність вхідних транзакцій. Одна з таких транзакцій, а саме видалення процесу з історії, включає внутрішній логічний файл і представляє собою залучення одного поля для зміни стану програми. Тому у нас є 1 FTR та 1 DET. Разом зовнішні транзакції (EO та EQ) залучають всі логічні групи даних, які в сумі складають 12 DET, залучаючи при цьому один внутрішній та один зовнішній логічні файли, тобто 2 FTR.

Загалом, згідно з таблицями оцінки зовнішніх вхідних та вихідних транзакцій, маємо низьку складність для перших і середню для других.

Таблиця 4.4

Складність у співвідношенні вхідної транзакції до DET

EI	1-4 DET	5-15 DET	16+ DET
0-1 FTR	Низьке	Низьке	Середнє
2 FTR	Низьке	Середнє	Високе
3+ FTR	Середнє	Високе	Високе

Таблиця 4.5

Складність у співвідношенні вихідної транзакції до DET

EO та EQ	1-5 DET	6-19 DET	20+ DET
0-1 FTR	Низьке	Низьке	Середнє
2-3 FTR	Низьке	Середнє	Високе
4+ FTR	Середнє	Високе	Високе

Підбиваючи підсумки усіх проведених досліджень з визначення складності графічного інтерфейсу та його взаємодії з користувачем за допомогою методу функціональних точок, можна зауважити, що програмна система має низьку складність у роботі з даними та файлами для їх зберігання. Це безперечно зменшує навантаження на інтерфейс. Оцінка транзакцій показала, що операції з вхідними даними є максимально оптимізованими та мають мінімальну складність. Проте зовнішні вихідні транзакції перебувають поблизу межі між низькою та середньою складністю, але все ж вказують на середню складність. Загалом, можна сказати, що мета створення графічного інтерфейсу була досягнута, оскільки як складність даних, так і методів введення параметрів є мінімальною. Водночас, надання відповідей програми та представлення звітів є детальним та насиченим.

4.3 Тестування програми

Для проведення тестування програми важливо скласти сценарії використання, які демонструватимуть роботу програми у різних умовах та форматах введення параметрів. Попередній розділ охопив роботу інтерфейсу загалом з усіма можливими параметрами. Тут розглянемо відмінності середовищ та об’єму завантаження на програму й порівняємо їх зі стандартними параметрами.

Перший сценарій (Рис. 4.3.1) розглядатимемо як стандартний. Загальна кількість файлів – 7, приблизно однакових за розміром – 1 хвилина. Запуск буде для всієї теки без обмежень за назвою файлів. Ключовими словами слугуватимуть як

часті випадки ("друзі", "коли"), так і рідкісні, що можуть зустрічатись у конкретній аудіокнизі ("Дракула").

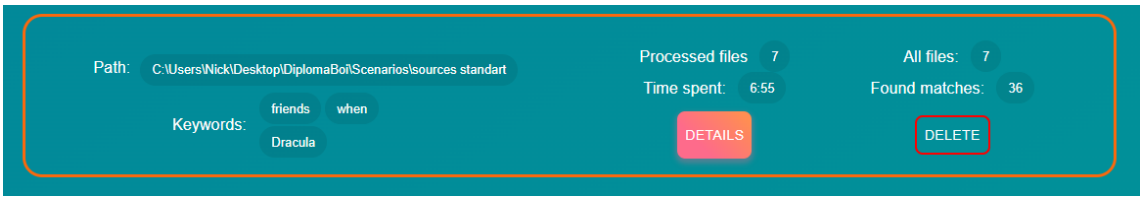

Рис. 4.3.1 Результати стандартного сценарію

Час виконання тестів складає 6 хвилин 55 секунд, і вони виявили 36 співпадінь.

Другий сценарій (Рис. 4.3.2) передбачає використання великої кількості файлів - 42, серед яких є 3 файли, які не можуть бути проаналізовані, для відображення зміни статистики. 28 файлів – це копії одного з невеликих файлів, але це не впливає на тестування, оскільки вони не зберігаються в пам'яті. А інші 11 файлів – взагалі не мають такого ключового слова. Цей сценарій використовує лише одне ключове слово - прізвище автора ("Кафка").

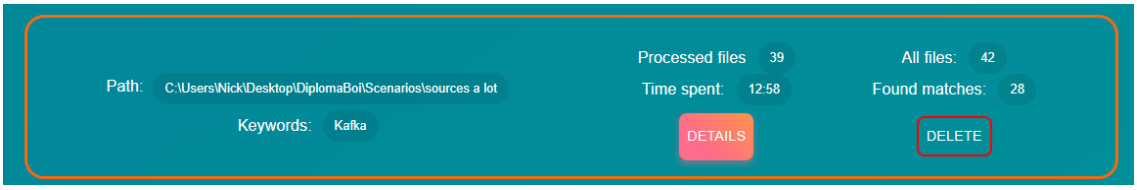

Рис. 4.3.2 Результати сценарію з великою кількістю файлів

Час виконання – 12 хвилин 58 секунд, при цьому знайдено лише 28 співпадінь, що трохи менше, ніж у стандартному сценарії.

Третій сценарій (Рис. 4.3.3) передбачає роботу з одним великим файлом - розширеною версією аудіокниги "Дракула", розміром 40 хвилин 22 секунди, що в десяток разів більше за окремі файли. Суть цього сценарію перевірити роботу з великим об’ємом даних. Алгоритм має розбивати цей файл на менші частинки і опрацьовувати їх окремо. У цьому сценарії також враховується пошук надзвичайно часто вживаних слів у книзі, наприклад, артиклів чи дієслів ("the", "have").

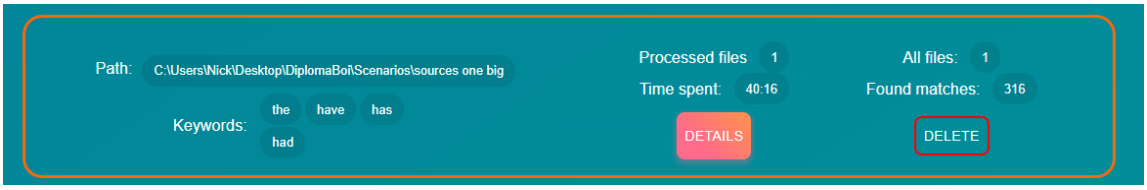

Рис. 4.3.3 Результати сценарію з 1 великим файлом

Час виконання процесу склав 40 хвилин 16 секунд, що на 6 секунд менше за розмір файлу, а отже можемо бачити роботу алгоритму статистичного аналізу на таких великих файлах, але знайдено 316 співпадінь з ключовими словами.

Тестування проводилося на комп'ютері із такими характеристиками: процесор 2.7 ГГц, 6 ядер, оперативна пам'ять 16 ГБ, 64-бітна операційна система Windows, диск M.2. Також важливо порівняти розроблений застосунок з відомими аналогами (Табл. 4.6).

Таблиця 4.6

Порівняння систем

Порівняння	Braina	Otter	PicoVoice	Розроблений застосунок
Розмір програми	340Мб + 2Гб	102Мб	Онлайн	105Мб
Методика транскрипції	Мовне моделювання	Мовне моделювання	Комбіноване (мовне та акустичне моделювання)	Мовне моделювання
Якість транскрипції	5/10	9/10	10/10	8/10
Швидкість пошуку до об’єму файлу	4:30 до 4:30 (паралельно)	0	0.003 до 0:49	4:30 до 4:30 та 40:16 до 40:22
Якість аналізу	5/10	8/10	7/10	9/10
Розмір контексту	0/10	7/10	5/10	9/10
Наявність звіту	3/10	9/10	8/10	10/10
Зручність інтерфейсу	3/10	8/10	8/10	9/10

4.4 Напрямки оптимізації та вдосконалення програми

Після ретельного аналізу програмної системи важливо виявити напрямки для її вдосконалення, щоб забезпечити подальший розвиток та вирішення наявних проблем.

Почнемо з оптимізації. Під час тестування програми на велику кількість ключових слів було помічено недоліки у взаємодії графічного інтерфейсу та серверної частини. При великому обсязі знайдених співпадінь в тексті виникали значні затримки завантаження та відображення списку у вікні браузера. Ефективним рішенням цієї проблеми буде використання віртуалізованого списку. Це технологія оптимізації великих обсягів даних у інтерфейсі, коли сервер передає лише частину даних та запитує нові при необхідності. Такий підхід не лише скоротить час очікування при великих обсягах даних, а й зменшить навантаження на обчислювальні ресурси.

Розвиток програм чи систем може відбуватися у двох напрямках: розширення функціоналу вже існуючих можливостей (ширина) або додавання нового функціоналу (довжина).

Для росту в ширину потрібно збільшувати підтримувані типи файлів і щоб програма сама їх конвертувала у зручний формат WAV. Також потреба полягає не в усіх форматах, а в тих що найбільше розповсюджені такі як MP3, MP4, FLAC. Та можливо запровадження використання унікальних типів файлів які зберігають звукову доріжку, але мають надзвичайну ефективність по використанню пам’яті.

Збільшення швидкості та якості розпізнавання слів у тексті є головною ціллю для зросту в довжину. Для вирішення цієї задачі потрібно збільшити навчання штучного інтелекту, а також робити спектральний аналіз звуку. Це дасть змогу надзвичайно чітко та швидко аналізувати аудіофайли. Проте для такого аналізу потрібно дослідити немало інформації щоб змогти його відтворити, а також автоматизувати у програмну систему

Висновки

У четвертому й останньому розділі ми представили вже наявні можливості системи та їхній вплив на користувача.

Проаналізували основні елементи інтерфейсу й показали його низьку складність, використовуючи метод оцінки функціональних точок. Таким чином, дані, які відображаються користувачу, виявилися низькоскладними, так само як і вхідні операції. Однак складність вихідних операцій виявилася помірною, що відповідає проектним вимогам до повних та корисних звітів.

Також було проведено тестування програми у різних сценаріях та при різних навантаженнях, враховуючи зміну кількості файлів, пошукових слів та їх розмірів. Крім того, важливо зазначити, що розроблена система не поступається аналогічним системам, а за деякими показниками їх навіть перевершує.

Також були визначені напрямки вдосконалення та оптимізації системи. Такими стали: розширення підтримуваних форматів файлів та вдосконалення швидкості та якості розпізнавання мови шляхом навчання штучного інтелекту та проведення спектрального аналізу вмісту аудіофайлу.

ВИСНОВКИ

Підсумовуючи роботу над дипломною роботою, слід відзначити досягнення поставленої мети та успішне виконання запланованих завдань.

Був проведений детальний аналіз методів та інструментів, які можуть допомогти в аналізі вмісту аудіофайлів, засобів, які стали основою для визначення цілей автоматизації, що були втілені у програмній системі. Робота включала в себе вирішення проблем, що існували у програм аналогів, оптимізацію алгоритму та урахування потреб користувачів. Було досліджено етапи та можливості машинного навчання для штучного інтелекту, а також використано отримані знання з дослідження у побудові та вирощенні штучного інтелекту для програмної системи. Також було продемонстровано важливість простоти графічного інтерфейсу через оцінку складності за допомогою методу функціональних точок, відповідаючи вимогам щодо швидкості пошуку та зручності використання.

Робота охоплювала дослідження різних методів розпізнання мовлення, контекстного пошуку та визначення вимог до отриманих результатів пошуку. Надані звіти будуть корисні та сприятимуть ефективному пошуку інформації у текстових файлах, завдяки ефективній системі збору інформації про знайдені співпадіння.

Був створений програмний продукт для аналізу вмісту аудіофайлів, підтверджено ефективність алгоритмічних рішень під час тестування програми. Роботу системи було протестовано на різних сценаріях та навантаженнях, виявивши переваги у порівнянні з іншими системами та підтвердивши результати досліджень з ефективності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке аудіофайли. 2023 (Електронний ресурс). – Режим доступу: https://www.easytechjunkie.com/what-are-audio-files.htm
2. 10 найчастіше використовуваних форматів аудіофайлу. 2023 (Електронний ресурс). – Режим доступу: https://www.makeuseof.com/tag/audio-file-format-right-needs/
3. Скінченні автомати - Кривий С. Л. "Скінченні автомати: теорія, алгоритми, складність" (книга)
4. Поняття аналізу тексту. 2017 (Електронний ресурс). – Режим доступу: https://www.opentext.com/what-is/file-analysis
5. Алгоритми пошуку рядка. 2020 (Електронний ресурс). – Режим доступу: https://habr.com/ru/post/111449/
6. Алгоритми пошуку рядка. 2019 (Електронний ресурс). – Режим доступу https://ukrbukva.net/8677-Algoritmy-poiska-podstroki-v-stroke.html
7. Основні поняття алгоритмів та структур даних - Н. Вірт. АЛГОРИТМИ І СТРУКТУРИ ДАНИХ (книга)
8. Програма Braina. 2023 (Електронний ресурс). – Режим доступу: https://www.brainasoft.com/braina/speech-to-text.html
9. Програма Otter. 2023 (Електронний ресурс). – Режим доступу: https://otter.ai/
10. Програма PicoVoice. 2023 (Електронний ресурс). – Режим доступу: https://picovoice.ai/
11. Реферат на тему «Поняття контексту слова» . 2017 (Електронний ресурс). – Режим доступу: https://vseosvita.ua/library/referat-na-temu-ponatta-kontekstu-ta-jogo-tipiv-179499.html
12. Багатопоточність. 2022 (Електронний ресурс). – Режим доступу: https://www.techtarget.com/whatis/definition/multithreading
13. Ліниве програмування. 2016 (Електронний ресурс). – Режим доступу: https://www.infoworld.com/article/3122803/the-power-of-lazy-programming.html
14. Машинне навчання 2023 (Електронний ресурс) – https://www.it.ua/knowledge-base/technology-innovation/machine-learning

15. Головний сайт CMU Sphinx 2023 (Електронний ресурс) – https://cmusphinx.github.io/

16. Відмінність між різними форматами дисків. 2021 (Електронний ресурс). – Режим доступу: https://photographylife.com/nvme-vs-ssd-vs-hdd-performance

17. Стаття про зручність графічного інтерфейсу. 2022 (Електронний ресурс). – Режим доступу: https://cases.media/article/golovne-pro-zruchnist-v-interfeisakh

18. Блог про важливість графічного інтерфейсу. 2022 (Електронний ресурс). – Режим доступу: https://uxdesign.cc/

19. Поняття функціональної точки. 2010 (Електронний ресурс). – Режим доступу: https://uk.theastrologypage.com/function-point

20. Аналіз функціональних точок . 2018 (Електронний ресурс). – Режим доступу: https://uk.wikipedia.org/wiki/Аналіз_функціональних_точок

21. Martin Kleppmann O`Rilley. Дизайн, програмні системи чутливі до даних: великі ідеї за масштабованими, надійними та керованими системами, 2019 – 320 - 420 с.

22. Ларман, Крег. Застосування UML 2.0 та шаблонів проектування: вступ до об’єктно-орієнтованого аналізу дизайну та ітеративної розробки — [3-е вид.] 2006. — 736с. (книга)

23. Девід Грифітс. Спочатку Котлін. Дружні настанови 2019 (книга)

24. Документація для фреймворку React. 2022 (Електронний ресурс). – Режим доступу: https://uk.reactjs.org/

25. Комбінація TypeScript та React 2023 (Електронний ресурс) https://www.typescriptlang.org/docs/handbook/react.html