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

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки”

на тему: Акустична модель системи розпізнавання українського мовлення

Виконав: студент 4 курсу, групи ТР-52

Сехін Олександр Петрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Стативка Юрій Іванович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Сехіну Олександру Петровичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Акустична модель системи розпізнавання українського мовлення”

керівник роботи доцент, к.т.н. Стативка Юрій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.

3. Вихідні дані до роботи результат розпізнавання української мови у звуковому сигналі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення для створення акустичної моделі, створити акустичну модель для української мови та систему яка буде надавати можливість розпізнавати українське мовлення .

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Мета та завдання роботи. 2. Опис системи автоматичного розпізнавання мови. 3. Опис засобів розробки. 4. Поетапне створення акустичної моделі для української мови. 5. Інтерфейс користувача та сценарій роботи користувача з

системою. 6. Висновки.

6. Публікації: _____

Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Сехін О.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стативка Ю.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи було створено вільну акустичну модель для української мови яку можна використовувати в системі розпізнавання мовлення. В роботі описується кожен етап побудови акустичної моделі для української мови за допомогою інструментів CMU Sphinx. Використовуючи створену модель була розроблена програма для розпізнавання української мови. Розроблена система працює в режимі реального часу та може розпізнавати звукові файли. Результат розпізнавання виводиться в текстове поле. Записка містить 63 сторінки 16 рисунків 4 формули та 17 посилань.

ABSTRACT

The purpose of the work was to create a free acoustic model for the Ukrainian language, which can be used in the speech recognition system. The paper describes each stage of building an acoustic model for the Ukrainian language using the CMU Sphinx tools. Using the created model, a program for recognizing the Ukrainian language was developed. The developed system works in real time and can recognize audio files. The result of the recognition is displayed in the text field. The note contains 63 pages of 16 figures, 4 formulas and 17 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

IBM (англ. International Business Machines) — американська компанія, один з найбільших в світі виробників і постачальників апаратного і програмного забезпечення.

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

MIDI (англ. Musical Instrument Digital Interface) — стандарт цифрового звукозапису на формат обміну даними між електронними музичними інструментами.

MFCC (англ. Mel-frequency cepstral coefficients) — є коефіцієнтами, які спільно складають MFC.

LPC (англ. Linear predictive coding) — спосіб перетворення аналогового сигналу в цифрову форму.

ВСТУП

Розпізнавання мови являє собою актуальну задачу, пов'язану з безліччю різних додатків, таких як, голосове керування, автоматична генерація субтитрів до відеоматеріалів, переклад аудіо записів в текст. Але в одно час це дуже складний процесс.

Системи розпізнавання мови складаються з безлічі компонентів, одним з яких є акустична модель. Розпізнавання мови неможливе без акустичної системи яка спів ставляє вимовлений звук з фонемою. На сьогоднішній день не існує вільної акустичної моделі для розпізнавання української мови.

Існуючі програми для розпізнавання мови такі як голосовий пошук Google або Yandex SpeechKit розпізнають мову шляхом відправки запиту на сервер, де він обробляється і результат розпізнавання відправляється назад користувачеві. Таким чином для використання цих програм необхідне постійне підключення до інтернету. Використовувати систему яка працює таким чином пов’язано з великими ресурсними затратами. А також є ймовірність перехоплення приватних даних. Тому представляється актуальним створення акустичної моделі для вільної системи яка не залежить від інтернету.

Створення акустичної системи є найскладнішим завданням при побудові системи розпізнавання мови. Так як для побудови якісної достовірної моделі будь-якої мови необхідно зібрати велику кількість акустичних даних із записами великого числа дикторів. Запис повинен проводиться в ізольованій від шуму кімнаті.

У роботі описується кожен етап розробки акустичної моделі для української мови, починаючи від вибору та огляду технологій для створення, налаштування необхідного програмного забезпечення та завершуючи описом використання системи майбутнім користувачем.

Зміст розділів даної роботи наступний:

У першому розділі описується постановка задачі для створення акустичної моделі та системи розпізнавання.

У другому розділі описуються архітектура систем розпізнавання.

У третьому розділі вказуються основні засоби розробки даної системи.

У четвертому розділі описано побудову нової акустичної моделі та її навчання.

У п’ятому розділі дано опис реалізованого програмного продукту, а також описано роботу користувача з програмною системою.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ АКУСТИЧНОЇ МОДЕЛІ ДЛЯ СИСТЕМИ РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ

Зараз існує велика кількість комерційних систем розпізнавання мови: Google, Amazon Alexa, IBM Watson, Siri, Yandex.

Вони є вільними для використання і пропонують відкриті API. Якість розпізнавання такими системами теж досить висока. Але вони також мають деякі недоліки.

Ці системи працюють лише через інтернет, відповідно, якщо немає мережі вони не працюють. Пересилання даних через мережу може вносити великі затримки при поганій якості зв'язку. Також існує можливість перехоплення приватних даних.

Метою роботи є створення акустичної моделі для української мови та системи яка буде надавати можливість розпізнавати українське мовлення за допомогою створеної акустичної моделі. Система повинна бути адаптована для різних дикторів в різних умовах.

Для досягнення поставленої задачі були сформульовані наступні завдання:

— створення фонетичного словника української мови;

— створити модель акустичних одиниць мови;

— зібрати достатню кількість звукозаписів українською мовою;

— зібрати текстового матеріалу для створення моделі мови;

— створити модель мови;

— навчити акустичну модель;

— розробити систему автоматичного розпізнавання мови;

Розроблена система повинна мати інтуїтивно зрозумілий інтерфейс користувача та забезпечувати наступні можливості:

— приймати звуковий потік з мікрофону;

— розпізнавати українські слова в реальному часі;

— вибирати звуковий запис збережений в операційній системі;

— розпізнавати звуковий запис;

— відображати вимовлені слова в текстовому полі;

— записувати та прослуховувати звуковий файл;

— розбивати звуковий файл по відрізкам тиші або часу;

Виконавши поставлені задачі в результаті буде отримано систему розпізнавання для української мови, яку можна використовувати в інших додатках. Ця система буде вільної та її можна буде використовувати незалежно від підключення до мережі.

2 ОПИС СИСТЕМИ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ МОВИ

Будь-яка архітектура статичної системи автоматичного розпізнавання мови складається з таких елементів [1]:

1. Модуль відділення корисного сигналу.
2. Акустична модель.
3. Мовна модель.
4. Декодер.

2.1 Акустична модель

Акустична модель — це файл, який містить статистичні уявлення кожного з окремих звуків, що складають слово. Кожному з цих статистичних уявлень присвоюється мітка, звана фонем. Звукова система української мови містить 38 звуків: 6 голосних і 32 приголосних. Таким чином, нам необхідно 38 різних фонем.

Акустична модель дозволяє оцінити розпізнавання сегмента мови з точки зору схожості на звуковому рівні. Для кожного звуку спочатку будується складна статистична модель, яка описує проголошення цього звуку в мові.

Акустична модель створюється шляхом створення великої бази даних мови і використання спеціальних навчальних алгоритмів для створення статистичних уявлень для кожної фонеми на мові.

Для акустичного моделювання мови використовуються приховані марковскі моделі (ПММ), при цьому кожна фонема є однією безперервною ПММ першого порядку. Модель фонеми має три стану: перше описує початок фонеми, друге представляє центральну частину і третє — кінцівку. ПММ слова виходить шляхом з'єднання в ланцюжок моделей фонем з відповідного фонетичного алфавіту.

Аналогічним чином з'єднуються моделі слів один з одним, утворюючи моделі фраз. Мета навчання акустичних моделей, заснованих на ПММ, полягає в тому, щоб за навчальною послідовністю спостережень визначити такі параметри моделі, з якими ймовірність появи цієї послідовності була максимальною [2]. Одним із способів вибору таких параметрів моделі, щоб локально максимізувати дану ймовірність, є метод Баума-Уелча. В ході процесу навчання визначається розподіл ймовірностей переходів між станами ПММ (матриця перехідних ймовірностей), розподіл ймовірностей появи символів спостереження в стані, розподіл усіх розподіл початкових станів моделі.

На виході даного блоку формується набір ПММ акустичних одиниць мови для прийнятого фонетичного алфавіту.

2.2 Опис прихованої марковської моделі

Марковська моделі застосовує кінцевий автомат, який містить N-станів, які називаються прихованими. Кожен перехід між станами в кожен дискретний момент часу t не є детермінованими. Цей перехід відбувається відповідно до законів ймовірності. Його описують за допомогою матриці ймовірностей переходів A_{NN}.

Діаграми переходів між станами ПММ зображена на рисунку 2.1.

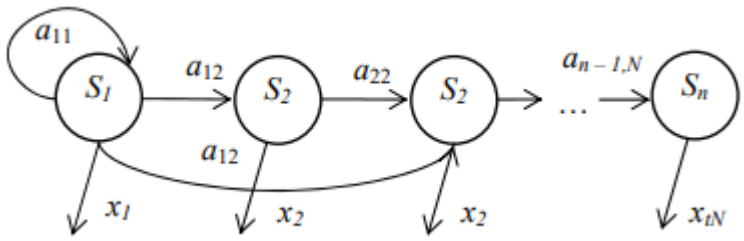

Рисунок 2.1 Структурна схема переходів ПММ

Знаходження моделі в деякому стані i відповідає певній стаціонарності спостережуваного сигналу на обмеженому часовому інтервалі. З'являється проста

фізична інтерпретація ПММ: розглядається процес, який іноді стрибкоподібно змінює свої характеристики [3].

При здійсненні чергового переходу в новий стан i в момент часу t відбувається генерація вихідного вектора x_t, званого параметричних вектором, відповідно до багатовимірної функцією розподілу ймовірностей $\mathrm{f}_j(x)$.

Результатом роботи прихованої марковської моделі є послідовність векторів $[(\{x)]\ 1, x_2, \ldots, x_T\}$ довжиною T. Перевагою ПММ є можливість обробки послідовностей і сигналів різної довжини, що утруднено при роботі з штучними нейронними мережами, зокрема [6].

Функція щільності ймовірностей $f_j(x)$ для стану j описується, як правило, зваженої гаусом сумішшю:

$$f_j(x) = \sum_{i=1}^{M} w_i p_i(x) \qquad (2.1)$$

де M — кількість компонент суміші; w_i — вага компонента суміші; $p_i(x)$ — нормальний розподіл для D-мірного випадку.

Робота з прихованими марковскими моделями, як і з будь-якої іншої адаптивної експертної системою, здійснюється в два етапи [4]:

1. Навчання — визначення параметрів моделі — алгоритм Баума-Велчє;

2. Визначення — наскільки ймовірним є те, що спостережувана послідовність векторів $[(\{x)]\ 1, x_2, \ldots, x_T\}$ була згенерована даною моделлю — алгоритм максимуму правдоподібності Вітербо. Далі наводиться короткий опис перерахованих вище алгоритмів.

3. Навчання прихованої марковської моделі. Процес навчання прихованої марковської моделі полягає у визначенні за допомогою набору навчальних зразків наступних параметрів:

— матриці ймовірностей переходів між станами A_{NN};

— параметрів гауссових сумішей (математичне очікування, матриця коваріації і ваги) для кожного стану.

Для вирішення цих завдань спільно застосовуються два ітераційних алгоритму: forward-backward і Baum-Welch re-estemation. В алгоритмі forward-

backward вводяться дві функції: прямого поширення ймовірності $a_j(t)$ і зворотного $\beta_j(t)$. Значення величини $a_j(t)$ являє собою ймовірність спостереження послідовності векторів $[(\{x)]\ 1, x_2, \ldots, x_t\}$ і знаходження ПММ в стані j в момент часу t:

$$a_j(t) = P\ (x_1, x_2, \ldots, x_t |\ statet\ =\ j) \tag{2.2}$$

Величини $a_j(t)$ і $a_j(t-1)$ пов'язані ітераційним виразом

$$a_j(t) = \left[\sum_{i=2}^{N-1} a_j(t-1)A_{ij}\right] f_j(x_t), \tag{2.3}$$

де A_{ij} — ймовірність переходу зі стану i в стан j; $f_j(x_t)$ — ймовірність спостереження вектора x_t в стані j. Зворотна функція $\beta_j(t)$ являє собою ймовірність знаходження ПММ в стані j в момент часу t з подальшим спостереженням послідовності $[(\{x)]\ t\ +\ 1, x_{t+2}, \ldots, x_T\}$:

$$\beta_j(t) = P\ (x_1, x_2, \ldots, x_t |\ statet\ =\ j). \tag{2.4}$$

Для якісного навчання прихованої марковської моделі потрібно безліч зразків сигналу: від декількох десятків до декількох сотень примірників. Також необхідно дотримуватися умова лінійної незалежності навчальних зразків, в іншому випадку, в процесі навчання відбувається виродження матриці коваріації, наслідком чого є повна непрацездатність моделі [7].

При практичній роботі з прихованими марковскими моделями доводиться вирішувати ряд ключових завдань:

1. Вибір системи параметричних векторів, наприклад, для розпізнавання мови використовуються кепстральних коефіцієнти (MFCC), коефіцієнти лінійного передбачення (LPC) та ряд інших;

2. Розробка алгоритму нормалізації параметричних векторів;

3. Вибір кількості станів моделі N і числа компонент гаусом суміші M;

4. Первісна сегментація навчальних векторів для знаходження наближених значень математичних очікувань гауссових сумішей на початковому етапі навчання і т.д.

Необхідно зауважити, що немає універсального алгоритму визначення перерахованих вище параметрів і в кожному конкретному випадку, в залежності від

розв'язуваної задачі, може знадобитися проведення величезної кількості експериментів, перш ніж будуть досягнуті необхідні результати точності розпізнавання.

При навчання є ймовірність виникнення ситуації, коли значення ймовірностей в знаменнику будуть мати значення близькі до нуля. Це призведе до переповнення регістрів процесора та виникненням помилок, які не дозволять продовжити процес. Через це зазвичай використовується логарифмічна арифметика.

2.3 Мовна модель

Мовна модель дозволяє визначити найбільш ймовірні словесні послідовності. Після того, як чергове слово було розпізнано, мовна модель визначає, наскільки це слово узгоджується з попередніми розпізнаними словами.

Мовна модель тренується за великим обсягом текстових даних (гігабайти текстів). За цими тренувальним даними обчислюється ймовірність появи слів в певному контексті, ґрунтуючись на частоті цих словосполучень.

Найпоширеніші мовні моделі використовують n-грамні мовні моделі — вони містять статистики послідовностей слів. Щоб отримати гарну точність, мовна модель повинна бути дуже успішна в обмеженні пошуку, тобто дуже хороша в прогнозі наступного слова. Мовна модель зазвичай обмежує запас слів. Це проблема для розпізнавання імен. Щоб впоратися з цим, мовна модель може містити невеликі частини, наприклад, підслова або навіть фонеми. Однак в цьому випадку обмеження пошуку зазвичай гірше і точність розпізнавання нижче, ніж з мовної моделлю, заснованої на словах. Ці три моделі разом складають движок розпізнавання мови. Найскладніша частина розпізнавання мови — зробити пошук збігів точним, і зробити його досить швидким.

2.4 Декодер

Декодер — це програмний компонент системи розпізнавання, який поєднує дані, одержувані в ході розпізнавання від акустичних і мовних моделей, і на підставі їх об'єднання, визначає найбільш ймовірну послідовність слів, яка і є кінцевим результатом розпізнавання злитого мовлення. Це компонент, який зіставляє вхідний мовний потік з інформацією, що зберігається в акустичних і мовних моделях, і визначає найбільш ймовірну послідовність слів, яка і є кінцевим результатом розпізнавання.

Мовний декодер прослуховує окремі звуки вимовленого користувача, а потім шукає відповідності ПММ в акустичної моделі.

Коли він знаходить відповідний ПММ в акустичній моделі, декодер приймає до відома фонему. Декодер відстежує відповідні фонеми, поки не досягне паузи в промові користувача.

Коли досягнута пауза, декодер шукає відповідні серії фонем, які він почув в своєму словнику вимови, щоб визначити, яке слово було вимовлено.

2.5 Етапи розпізнавання мови

Першим етапом процесу розпізнавання мови є визначення границь мовлення у звуковому сигналі, що надходить від мікрофона. Для цього використовується властивість відмінності значень енергії для мовних сегментів сигналу і для сегментів фонового шуму. Потім здійснюється сегментація мовного сигналу і витяг ознак.

Метод передачі маркерів визначає проходження можливих шляхів по станам об'єднаної ПММ. На початок кожного слова зі словника ставиться маркер і застосовується ітеративний алгоритм оптимізації Вітербо, при цьому на кожному кроці зсувається маркер і для нього обчислюється імовірнісна оцінка по акустичної

та мовної моделі. Після обробки всієї послідовності векторів спостережень вибирається маркер, що має найбільшу ймовірність. Коли найкращий маркер досягає кінця оброблюваного сигналу, то шлях, яким він проходить через мережу, відомий, і з маркера зчитується послідовність пройдених слів, яка і є гіпотезою розпізнавання фрази. Крім того, може бути отримано кілька найкращих маркерів, таким чином створюється список кращих гіпотез фрази яка була виголошена.

Далі параметри мовлення надходять в основний блок системи розпізнавання — декодер.

Висновки до розділу

У даному розділі було розглянуто, що собою являє і з яких елементів складається система розпізнавання мови. А також етапи розпізнавання.

3 ОГЛЯД ІНСТРУМЕНТАРІЮ ДЛЯ СТВОРЕННЯ АКУСТИЧНОЇ МОДЕЛІ ТА СИСТЕМИ РОЗПІЗНАВАННЯ МОВЛЕННЯ

В даному розділі розглядаються програмні засоби та бібліотеки які необхідні для створення акустичної моделі для української мови та системи розпізнавання.

3.1 Набір інструментів CMU Sphinx

Набір інструментів CMU Sphinx надає програмні засоби для створення необхідних моделей для систем розпізнавання мови. Для розробки акустичної моделі та системи розпізнавання необхідні пакети Sphinxbase, Sphinx4 та Sphinxtrain

Sphinxbase — надає необхідні бібліотеку для Sphinxtrain.

Sphinx4 — надає програмні інструменти для написання системи розпізнавання на мові програмування Java.

Sphinxtrain — надає інструменти для навчання акустичним моделям.

У 2000 компонент системи розпізнавання мови Sphinx 2 був опублікований як відкритий код, а в 2001 компонент Sphinx 3. На даний момент активно розробляється Sphinx 4 (написаний на Java, зручний для вбудовування в серверні системи) і PocketSphinx (написаний на C і зручний для вбудованих систем). Поширюється під ліцензією BSD.

Sphinx - це дикторонезалежної распознаватель безперервної мови, який використовує Приховані Марковские моделі і n-програмних статистичну мовну модель. Sphinx має можливості розпізнавання тривалої мови, дикторонезалежної величезний словник розпізнавання. Sphinx4 повний і переписаний мовної движок Sphinx, головна мета якого забезпечити гнучкий каркас для дослідження в розпізнаванні мови. Sphinx4 написаний повністю на мові програмування Java. Sun

Microsystems внесла великий внесок в розвиток Sphinx4 і допомогу в програмної експертизі проекту, що і пояснює вибір мови програмування, на якому написана система.

Кожна система Sphinx складається з двох компонентів: тренера і декодера. Тренер необхідний для створення акустичної моделі, адаптованої під конкретні потреби, а декодер виконує власне розпізнавання. Тренер Sphinx виконує побудову акустичної моделі. Необхідно добре розуміти як влаштована система розпізнавання мови, щоб правильно використовувати тренер Sphinx. Можливість роботи в такому режимі дуже корисна при створенні загальнодоступних сервісів, наприклад, автоматизованих телефонних служб і т.д.

3.2 Середовище розробки IntelliJ IDEA 2018

Програмне забезпечення JetBrains IntelliJ IDEA - це провідна середовище швидкої розробки на мові Java. IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторинга коду, вбудовану підтримку технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами управління версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів.

Середовище розробки IntelliJ IDEA дозволяє ефективно, якісно і швидко писати код, при цьому не втрачаючи уваги з контексту поточного файлу. За допомогою IntelliJ IDEA можливо легко переглянути структуру виклику та пов’язані функції, повернені значенні та стан тестування[5]. На сьогоднішній день продукти JetBrains з сімейства IntelliJ IDEA містять в собі багатий інструментарій. У сімействі IntelliJ IDEA є IDE, мультиплатформний редактор коду Visual Studio Code, який доступний для систем на базі операційних систем Windows, Mac та Linux. Також

ведеться розробка рішення для користувачів під платформою Mac OS – IntelliJ IDEA for Mac.

Для IntelliJ IDEA є документація досить великого обсягу англійською мовою. Під IntelliJ IDEA розроблено безліч plug-ins (розширень), що дозволяють помітно розширювати її можливості і функціональність. Доступ до них здійснюється в налаштуваннях системи в пункті Plugins.

3.3 JavaFx

JavaFX — це платформа яка використовується при розробці API, вона надає інструменти для створення додатків для персонального комп'ютера, чи інших електроних пристроїв. Технологія JavaFX полегшує розробку графічного інтерфейсу користувача.

Технологія JavaFX була вперше продемонстрована корпорацією Sun Microsystems на JavaOne, міжнародної конференції розробників, в травні 2007. Платформа JavaFX 1.0 була анонсована 4 грудня 2008 року.

Починаючи з версії Java 11 більше не входить в Java SE і не розробляється компанією Oracle (як окремий модуль підтримується компанією Gluon). Але Oracle буде вносити необхідні зміни до 2022 року як для частини Java SE 8.

JavaFX використовує декларативну мову програмування JavaFX Script для створення графічного інтерфейсу. Необхідно встановити JavaFX SDK, щоб використовувати цю мову. Це дає змогу одночасно використовувати мову програмування JavaFX Script для створення графічного інтерфейсу користувача та мову Java для проектування логіки програмного додатка.

3.4 Бібліотека Java.Sound.Sampled

Java Sound API була частиною стандартної бібліотеки Java 2 Platform з 1,3 релізі. Знайдений в javax.sound.sampled пакет Java Sound API забезпечує підтримку для відтворення і захоплення аудіо. Крім того, бібліотека пропонує програмний мікшер аудіо і MIDI (Musical Instrument Digital Interface) пристрої доступу. До цієї ради, ви дізнаєтеся, як захоплення аудіо через Java Sound API і відтворювати його.

Javax.sound.sampled пакет складається з восьми інтерфейсів, дванадцять верхнього рівня класу, дванадцять внутрішніх класів, а за двома винятками. Для запису і відтворення аудіо, вам необхідно мати справу тільки з цілому сім частин пакета.

WavFileClass бере на себе весь файл IO для читання і запису звукових даних і з звукових файлів. WavFileClass надає методи для доступу до даних WAV файлів з використанням різних Java примітивних типів, вони є long, int і double. Користувач може вибрати правильний тип для вирішення зразка даного файлу WAV.

Стандартні дозволи файлів WAV наведені в таблиці нижче. Для всіх дозволів, до 32-розрядних без знака, int хорошо підходить обробка вибірок з використанням типу. Для великих дозволів long следует використовувати тип. WavFileClass приймає на себе всі зразки дозволу 8 біт і менше непідписані, всі дозволи більше 8 біт підписуються (це, здається, стандарт для WAV - файлів).

Тип double може бути використаний для будь-якого дозволу даних WAV файлів. WavFileClass буде автоматично масштабувати значення вибірки і з -1,0 до 1,0 діапазону з плаваючою точкою.

Через процесу масштабування читання wav-файлів як подвійних, а потім їх виписування може привести до зміни деяких значень вибірки. Цей процес ілюструється на малюнку нижче. Цілі числа зі знаком, закодовані з використанням доповнення до двох, мають різні максимальні величини для позитивних і негативних чисел. Наприклад, 16-бітове значення зі знаком має діапазон від -32 768 до 32 767. При масштабуванні від цілого до подвійного необхідно використовувати

максимальну негативну величину, щоб жоден із зразків не мав значення менше 1,0. При масштабуванні від подвійного до цілого числа позитивна величина використовується для забезпечення того, щоб цілочисельний діапазон не перевищує.

Висновки до розділу

В даному розділі були розглянуті основні програмні засоби та бібліотеки для розробки системи. Був розглянутий пакет інструментів CMU Sphinx для створення акустичної моделі.

4 СТВОРЕННЯ АКУСТИЧНОЇ МОДЕЛІ ДЛЯ УКРАЇНСЬКОЇ МОВИ

В даному розділі розглянуті всі необхідні етапи побудови акустичної моделі для української мови.

4.1 Установка необхідних програмних інструментів

Для початку роботи з інструментами CMU Sphinx їх необхідно встановити на робочий комп'ютер, це можна зробити з офіційного сайту за таким посиланням https://cmusphinx.github.io/wiki/download/.

Також нам знадобиться середовище розробки Visual Studio 2017. Його можна скачати з офіційного сайту за таким посиланням https://visualstudio.microsoft.com/ru/vs/.

Після встановлення необхідних пакетів CMU Sphinx їх необхідно розпакувати та скомпілювати. Для цього необхідно запустити sphinxbase.sln в каталозі sphinxbase за допомогою Visual Studio 2017 та скомпілювати всі проекти. Так само необхідно скомпілювати пакет sphinxtrain. Після компіляції MS Visual Studio створить виконувані файли і бібліотеки які будуть знаходитись за посиланням sphinxtrain\bin\Release\x64\ та sphinxbase\bin\Release\x64\.

Також нам знадобляться пакети розробки Python та Perl. Python можна встановити з офіційного сайту за таким посиланням https://www.python.org/downloads/. Офіційний сайт Perl знаходиться за таким посиланням https://www.perl.org/.

Щоб переконатися, що python та perl були правильно встановлені в операційну систему, необхідно у командному рядку Windows ввести запити python –V та perl –v

відповідно. На ці запити командний рядок повинен показати версії встановлених компонентів (рис 4.1).

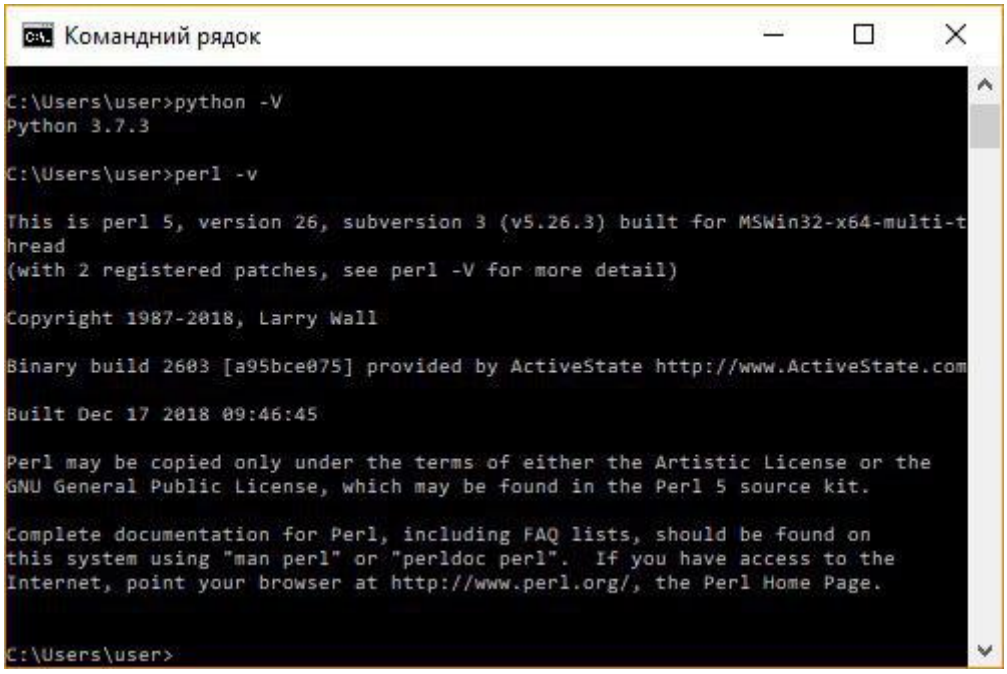

Рисунок 4.1 — Перевірка встановлення Python та Perl

Команди виконались успішно, а це означає що пакети розробки Python та Perl готові до використання.

4.2 Створення словника з українськими словами.

Словник повинен містити всі слова української мови, які в подальшому необхідно буде розпізнавати. Акустична система буде розуміти тільки ті слова, які знаходяться в словнику.

Для того щоб почати створювати словник, необхідно відкрити будь-який текстовий редактор та налаштувати параметри створюваного текстового файлу. Текстовий файл повинен мати наступні параметри:

— Формат кінця строки типу unix;

— Кодування файлу UTF-8;

— Формат самого файлу .dict;

Після встановлення параметрів, можна починати заповнювати словник необхідними словами. Необхідно пам’ятати що кожне слово повинно починатися з нового рядка.

Приклад створеного словнику:

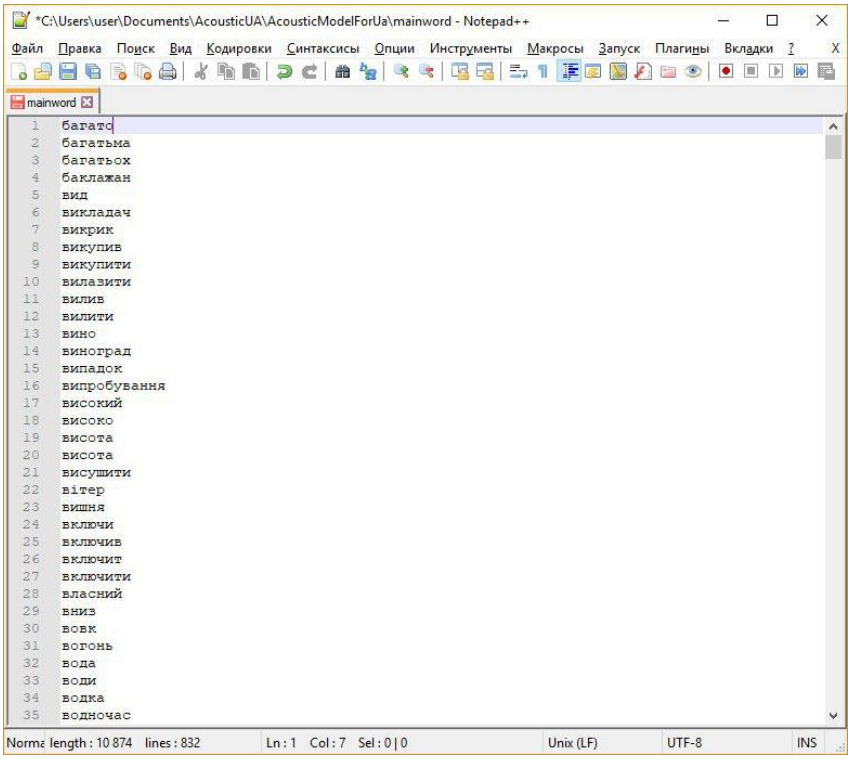

Рисунок 4.2 — Приклад створення словника

Для акустичної моделі було створено словник який містить понад 1000 українських слів.

4.3 Транскрибування словника.

Транскрибування словника представляє собою запис кожного слова у вигляді фонем. Для того щоб це зробити було створено програмний файл для автоматизації цього процесу. Ця програма була написана на мові програмування python.

Ця програма на вхід приймає текстовий файл в якому написані українські слова по одному в кожному рядку. Потім кожне слово аналізується і по правилам української мови представляє його у вигляді набору фонем. Кожна фонема української мови представляється у вигляді символів латинської мови. Так наприклад, українському звуку «ж» відповідає запис «zh», для звуку «й» відповідає запис «j» так далі.

Створений скрипт назвемо dict2transcript. Він повинен мати розширення .pl та бути збереженим в операційній системі в доступному місці.

Щоб застосувати цей скрипт треба в командному рядку ввести команду dict2transcript.pl попередньо вказавши шлях до цього файлу, та після пробілу вказавши шлях до файлу який потрібно транскрибувати та ім’я файлу який буде створено після транскрибування.

Приклад застосування скрипта:

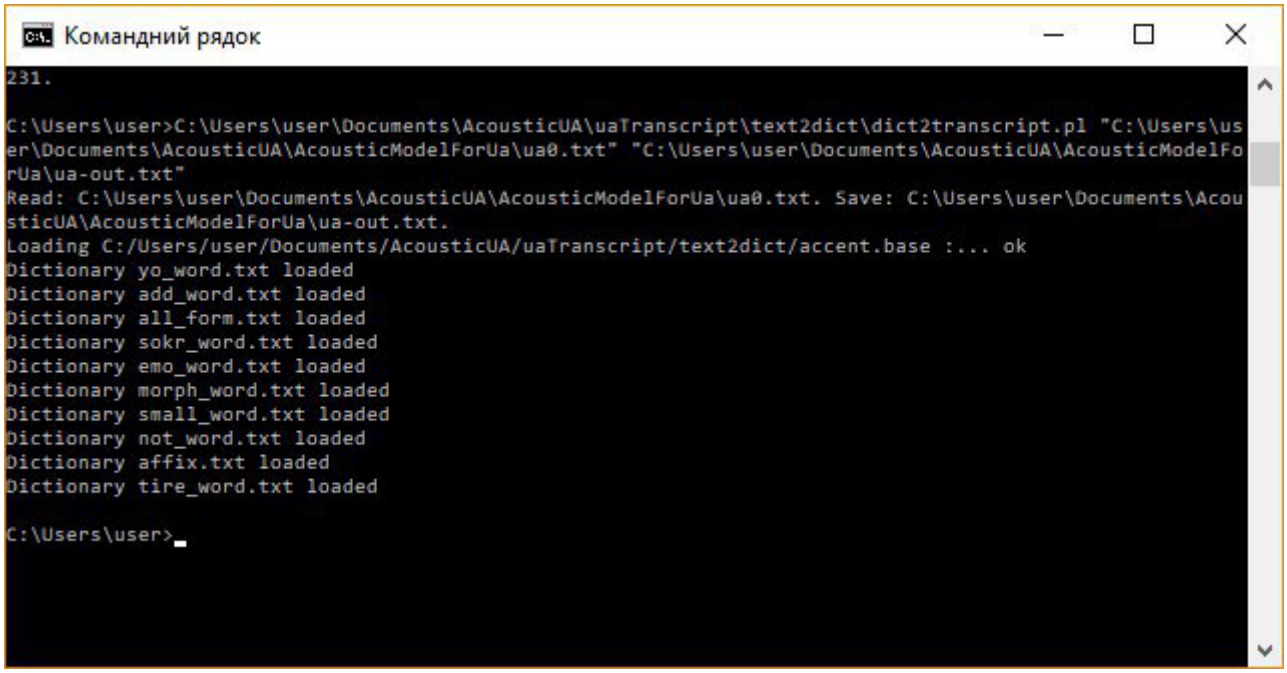

Рисунок 4.3 — Приклад застосування скрипта

Буде створено файл в форматі .txt. Для того щоб акустична модель розуміла його як фонетичний словник, формат треба змінити на .dict.

Приклад текстового файлу після транскрибування:

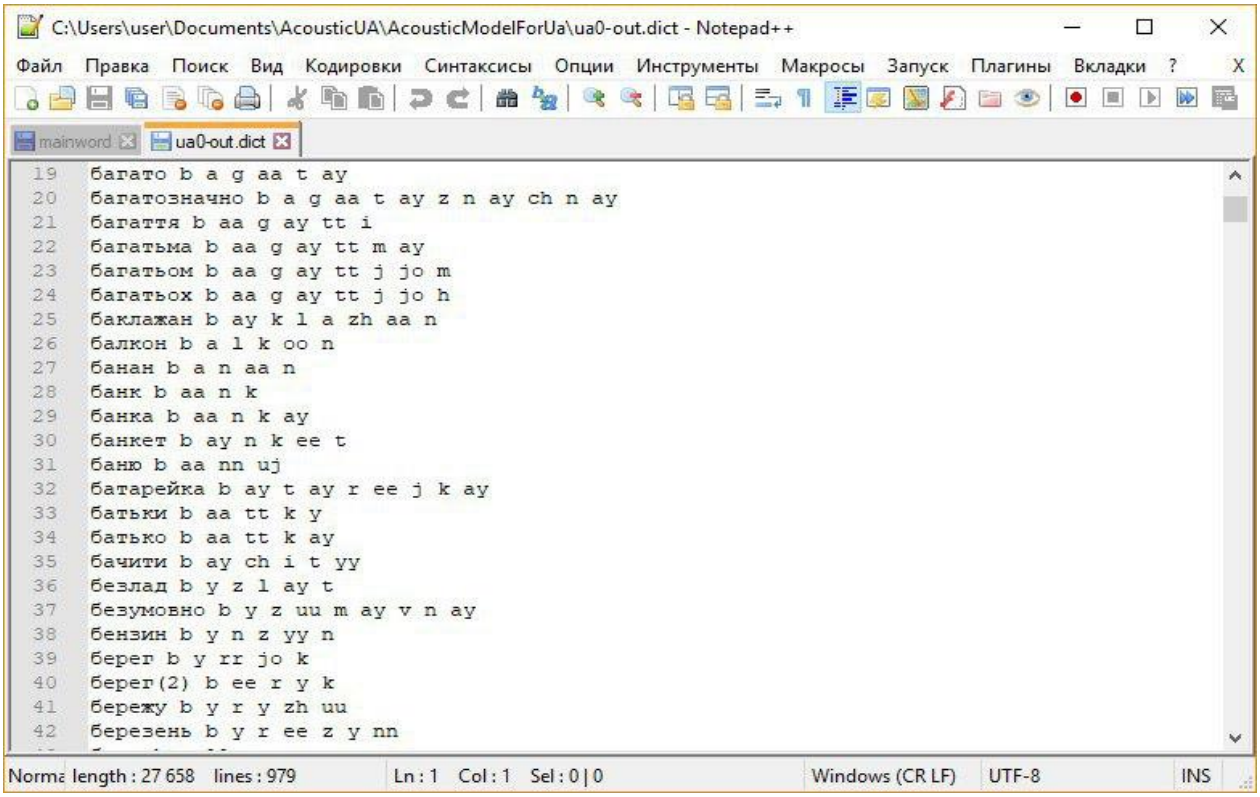

Рисунок 4.4 — Приклад транскрибування словника

Цей файл необхідний для того щоб представляти набір фонем отриманих після акустичної моделі у вигляді українських слів.

4.4 Файл з фонемами

Після того як фонетичний словник було створено, необхідно переписати всі фонеми в окремий файл. Файл з фонемами повинен містити одну фонему на рядок. Кількість фонем повинно відповідати фонемам, які використовуються в словнику.

В нашому випадку було використано 45 фонем які описують всі фонеми української мови, а також спеціальну фонему SIL яка характеризує тишу. Ця фонема необхідна для того щоб акустична модель могла розпізнавати тишу в звуковому файлі.

Приклад файлу з фонемами:

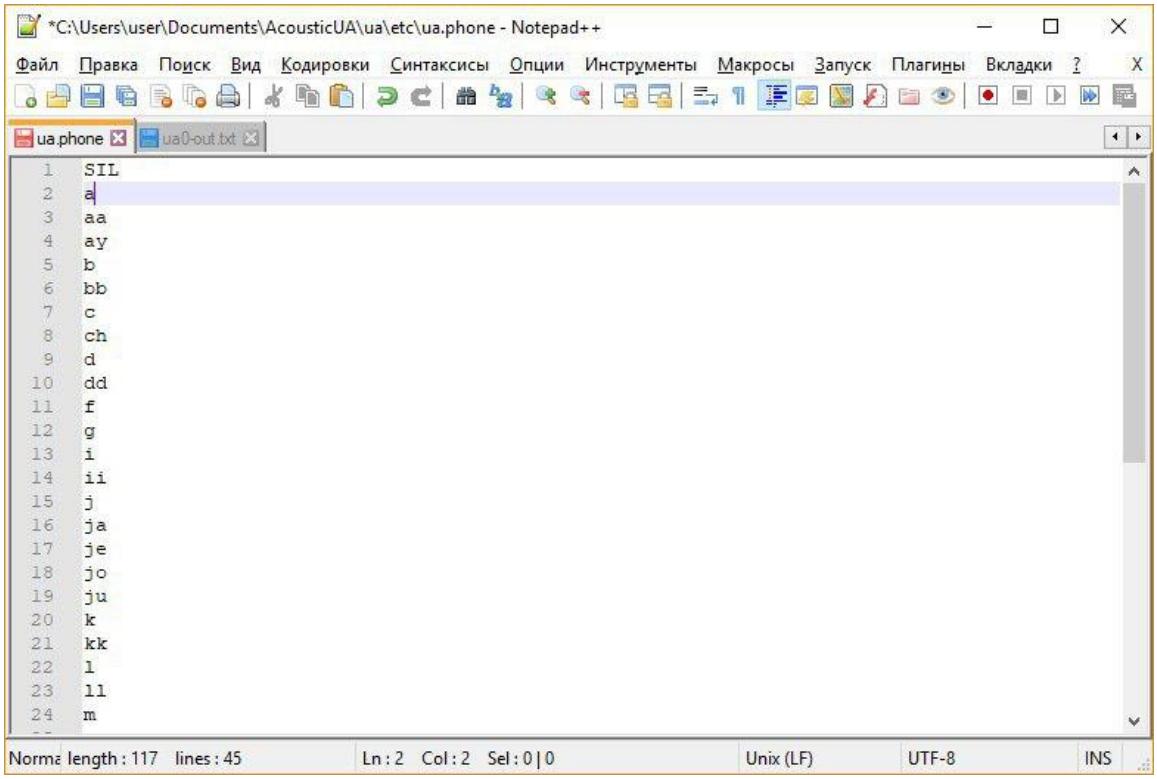

Рисунок 4.5 — Файл з фонемами

4.5 Навчальна база даних акустичної моделі

Акустична модель повинна спів ставляти звукові сигнали з необхідними фонемами. Для того щоб досягти цього необхідно створити велику навчальну базу даних.

База даних містить інформацію, необхідну для отримання статистики з промови у вигляді акустичної моделі. Для цього потрібно створити файл, який називається файлом розшифрування (Рисунок 4.6) та відповідний звуковий файл (Рисунок 4.7).

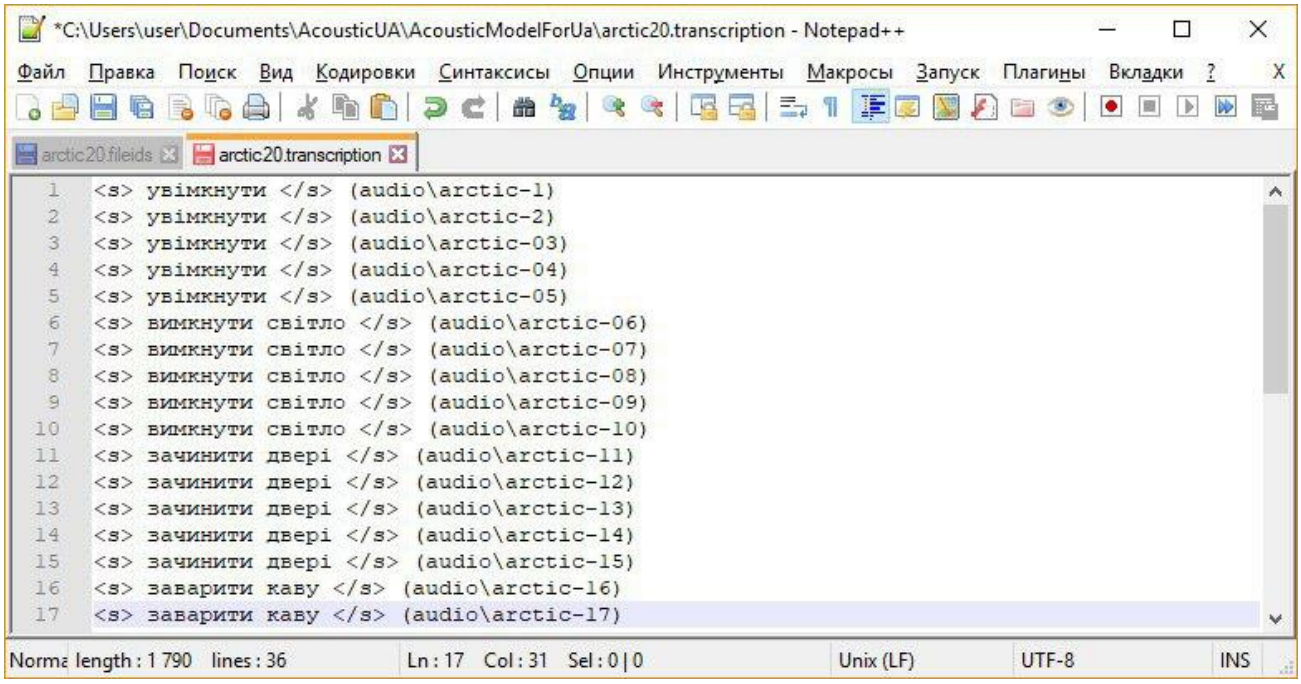

Рисунок 4.6 — Файл розшифрування

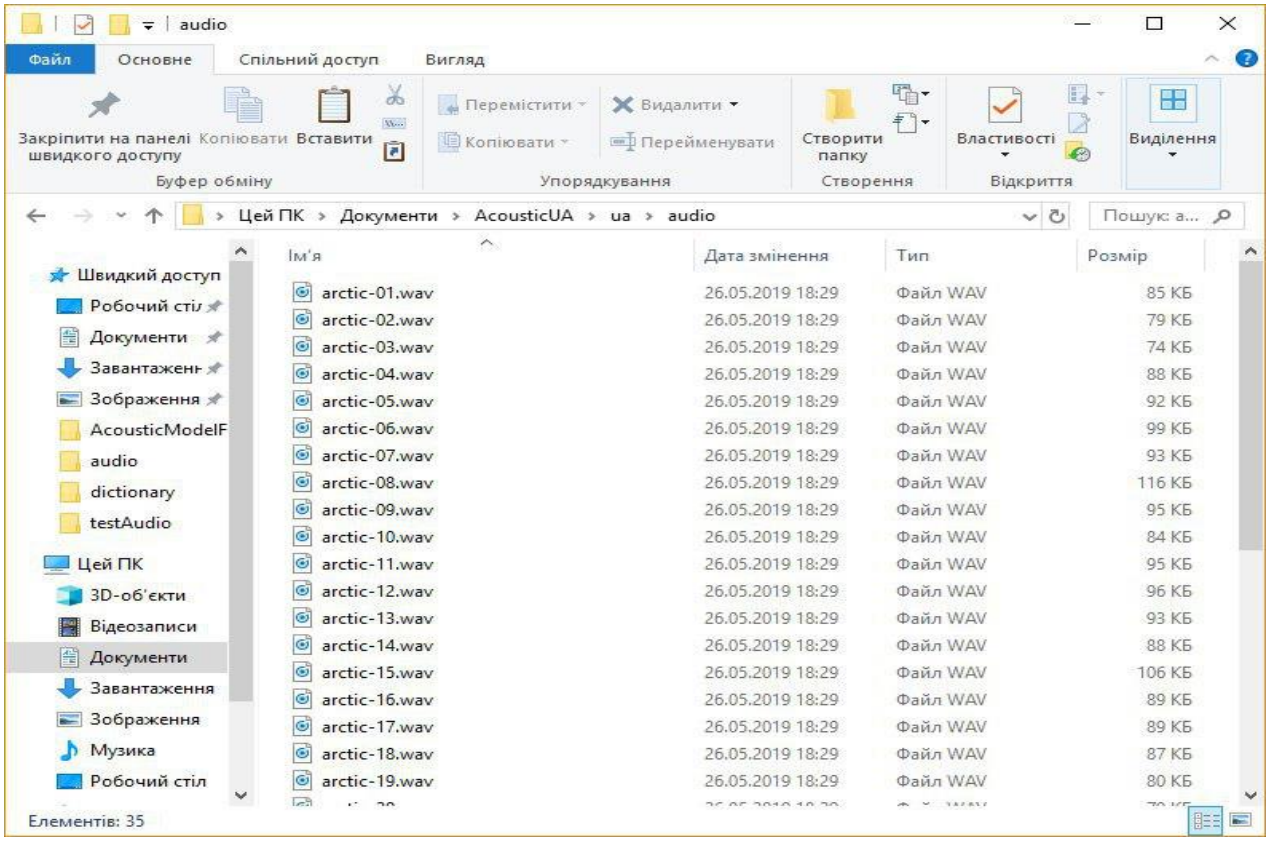

Рисунок 4.7 — Звукозаписи для навчальної бази даних

Файл розшифровки містить послідовність слів і немовних звуків в тому ж порядку, в якому вони зустрічаються в звуковому сигналі, за яким слідує тег, який використовується для зв'язування цієї послідовності з відповідним звуковим сигналом.

База даних повинна бути хорошим уявленням про те, яку мову треба буде розпізнавати. Наприклад, якщо потрібно розпізнавати телефонну мова, краще використовувати телефонні записи. Мова істотно розрізняється по різних каналах запису. Трансляція новин відрізняється від телефонного дзвінка. Мова, декодована з mp3, значно відрізняється від запису з мікрофона. Однак, якщо недостатньо мови, записаної в необхідних умовах, слід використовувати будь-яку іншу мову, яка є. Іноді має сенс передавати через телефонний кодек, щоб вирівняти звук. Часто можна додати шум і до тренувальних даними.

База даних повинна містити записи з достатньою кількістю мовців, з різними умовами запису, достатньою кількістю акустичних варіацій і всіма можливими лінгвістичними пропозиціями. Розмір бази даних залежить від складності завдання, яку потрібно виконати.

База даних повинна мати дві частини: навчальну частину і тестову частину. Зазвичай тестова частина становить приблизно 1/10 від загального обсягу даних.

4.6 Створення звукових записів

Аудіо записи повинні містити навчальне аудіо, яке повинно відповідати аудіо, яке ви хочете розпізнати в кінці. У разі невідповідності можна спостерігати значне зниження точності. Це означає, що якщо потрібно розпізнати безперервну мову, навчальна база даних повинна записувати безперервну мова. Для безперервної мови оптимальна тривалість аудіо запису становить від 5 до 30 секунд. Дуже довгі записи роблять навчання набагато складніше. Якщо потрібно розпізнавати короткі ізольовані команди, навчальна база даних також повинна містити файли з

короткими ізольованими командами. Краще проектувати базу даних так, щоб вона розпізнавала безперервну мова з самого початку і не витрачала час на команди [].

Файли аудіо записів повинні бути в форматі MS WAV з певною частотою дискретизації — 16 кГц, в режимі моно, з одним каналом. Дуже важливо, щоб аудіо файли мали певний формат.

Якщо аудіо записи не будуть відповідати потрібному формату, то акустична модель не буде навчена належним чином.

4.7 Тренер акустичної моделі

Тренер вивчає параметри для моделей звукових одиниць, використовуючи набір зразків мовних сигналів.

Тренеру потрібно повідомити, для яких звукових одиниць треба дізнатися параметри і послідовність, в якій вони зустрічаються в кожному звуковому сигналі у тренувальній базі даних. Ця інформація надається тренеру через файл розшифрування.

Таким чином, на додаток до звуковим сигналам з мовою і файлу транскрипції, тренеру необхідно отримати доступ до двох словників: одному, в якому допустимі слова в мові порівняно з послідовностями звукових одиниць, і іншому, в якому немає — мовні звуки зіставляються з відповідними немовними або мовно подібними звуковими одиницями. Потім тренер вивчає словники, щоб вивести послідовність звукових одиниць, пов'язаних з кожним сигналом і транскрипцією.

Як тільки всі необхідні файли були створені, можна починати тренування акустичної моделі. Для цього було використано пакет інструментів Sphinxtrain.

Щоб почати навчання, потрібно перейти в папку де знаходяться всі файли для бази даних і виконати наступну команду: python ../sphinxtrain/scripts/sphinxtrain -t an4 setup, де замість «an4» вказати ім’я папки з файлами бази даних (Рисунок 4.8).

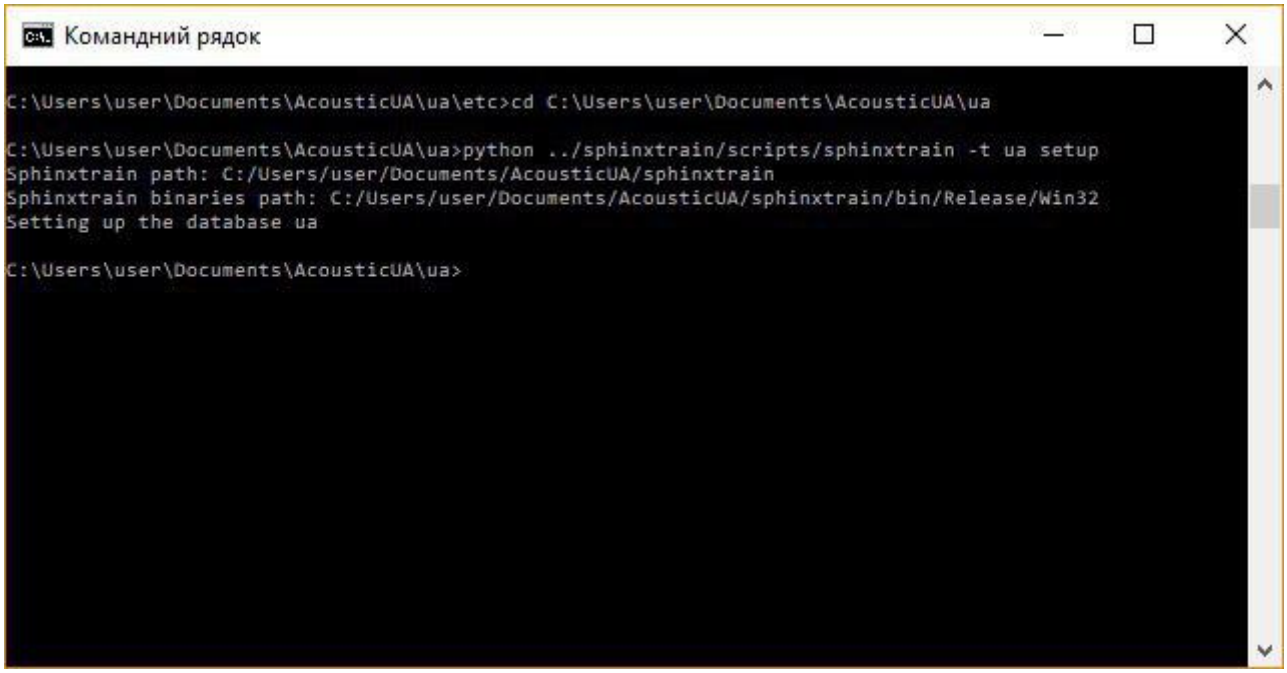

Рисунок 4.8 — Підготовка бази даних до навчання

Ця команда не навчить акустичну модель, а лише скопіює всі необхідні файли конфігурації в папку з файлами бази даних і підготує базу даних для навчання.

В процесі навчання будуть створені інші папки з даними, необхідними для навчання.

Після цієї настройки нам необхідно відредагувати файл конфігурації який буде створено. Він знаходиться в папці з файлами бази даних і його ім’я sphinx_train.cfg. У файлі конфігурації треба змінити деякі зміні, щоб налаштувати його для конкретної бази даних. Перш за все треба вказати який розмір має база даних. Для цього потрібно змінити команду $CFG_FINAL_NUM_DENSITIES. Це значення — кількість навчених в моделі сенонів. Чим більше сенонів у моделі, тим точніше вона розрізняє звуки. Також необхідно змінити команду $DEC_CFG_MODEL_NAME де вказати шлях до файлів акустичної моделі.

Тепер можна починати тренування акустичної системи. Для цього виконаємо команду python ../sphinxtrain/scripts/sphinxtrain run. Після цього буде виконано всі необхідні етапи. Якщо всі параметри було правильно налаштовано, в результаті буде отримано навчену акустичну модель.

Після тренування необхідно запустити декодер, щоб перевірити результати навчання. Декодер бере модель, тестує частину бази даних і довідкові записи і оцінює якість моделі. На етапі тестування використовується мовна модель з описом можливого порядку слів у мові.

Висновки до розділу

У даному розділі були розглянуті основні етапи побудови акустичної моделі для будь-якої мови. В даному випадку було описано створення акустичної моделі для української мови. Було показано як створити всі необхідні файли для акустичної моделі та як її навчити.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі описується взаємодія користувача з системою для розпізнавання мовлення, яка надає можливість розпізнавання як в реальному часі так і розпізнавання аудіо файлу. Система приймає на вхід акустичну модель для українського мовлення і проводить процес розпізнавання.

5.1 Розпізнавання українського мовлення в реальному часі

Система для розпізнавання української мови була написана за допомогою мови програмування Java та з використанням JavaFX Scene Builder 2.0. Дану систему можна використовувати в складі більших проектів.

Перш ніж почати роботу з програмою спочатку необхідно налаштувати параметри мікрофону в операційній системі. Він повинен мати частоту дискредитації 44100 Гц, розмір вибірки 16 біт, бути налаштований на другий канал та режим моно. Після чого можна починати процес розпізнавання мови натиснувши на кнопку «Розпочати розпізнавання» (Рисунок 5.1).

Система завантажить акустичну модель, фонетичний словник та мовну модель, проаналізує їх на наявність помилок та почне зчитування даних з мікрофону. Результат розпізнавання буде виводитись в реальному часі (Рисунок 5.2).

Щоб завершити розпізнавання потрібно в консолі ввести команду exit.

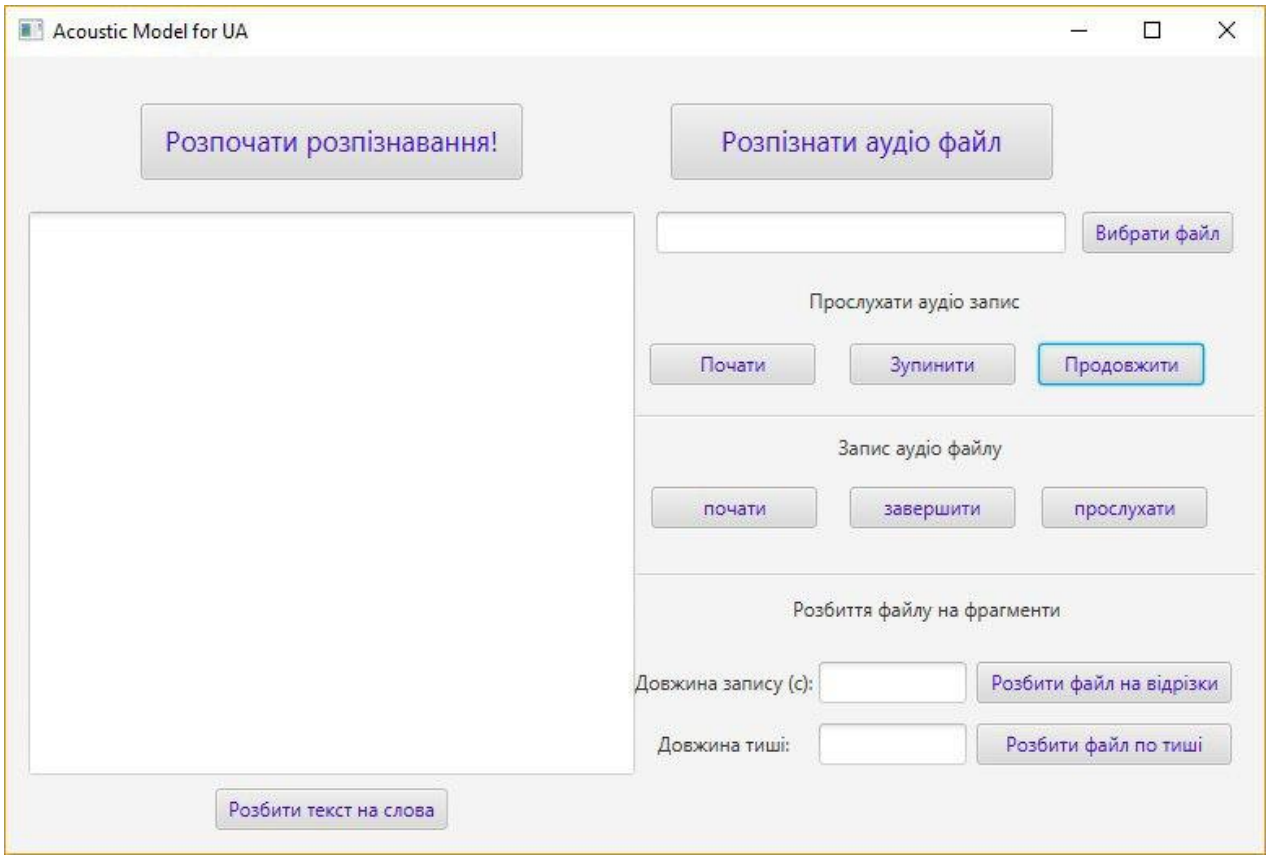

Рисунок 5.1 — Інтерфейс програми

```
08:30:05.167 INFO liveCMN            11.75 -0.08 0.21 -0.11 -0.43 -0.22 -0.19 -0.29 -0.30 -0.26 -0.13 -0.12 -0.17
08:30:05.347 INFO speedTracker          This  Time Audio: 0,96s  Proc: 0,90s  Speed: 0,94 X real time
08:30:05.347 INFO speedTracker          Total Time Audio: 8,59s  Proc: 19,37s 2,26 X real time
08:30:05.347 INFO memoryTracker         Mem  Total: 998,00 Mb  Free: 465,97 Mb
08:30:05.347 INFO memoryTracker         Used: This: 532,03 Mb  Avg: 593,92 Mb  Max: 743,00 Mb
08:30:05.347 INFO trieNgramModel     LM Cache Size: 2215 Hits: 583930 Misses: 2215
баклажан

08:30:07.710 INFO liveCMN            11.83 0.05 0.14 -0.16 -0.39 -0.21 -0.22 -0.27 -0.29 -0.26 -0.13 -0.13 -0.18
08:30:08.020 INFO speedTracker          This  Time Audio: 0,81s  Proc: 0,73s  Speed: 0,90 X real time
08:30:08.020 INFO speedTracker          Total Time Audio: 9,40s  Proc: 20,10s 2,14 X real time
08:30:08.020 INFO memoryTracker         Mem  Total: 998,00 Mb  Free: 379,50 Mb
08:30:08.020 INFO memoryTracker         Used: This: 618,50 Mb  Avg: 595,81 Mb  Max: 743,00 Mb
08:30:08.020 INFO trieNgramModel     LM Cache Size: 2239 Hits: 594199 Misses: 2239
багато

08:30:11.000 INFO liveCMN            12.00 0.10 0.04 -0.10 -0.41 -0.22 -0.23 -0.24 -0.30 -0.26 -0.13 -0.13 -0.19
08:30:11.297 INFO speedTracker          This  Time Audio: 0,91s  Proc: 0,89s  Speed: 0,98 X real time
08:30:11.297 INFO speedTracker          Total Time Audio: 10,31s  Proc: 21,00s 2,04 X real time
08:30:11.297 INFO memoryTracker         Mem  Total: 998,00 Mb  Free: 508,64 Mb
08:30:11.297 INFO memoryTracker         Used: This: 489,36 Mb  Avg: 588,21 Mb  Max: 743,00 Mb
08:30:11.297 INFO trieNgramModel     LM Cache Size: 2256 Hits: 604506 Misses: 2256
багатьом
```

Рисунок 5.2 — Розпізнавання в реальному часі

5.2 Розпізнавання аудіо запису

Інша можливість системі це розпізнавання аудіо запису з українським мовленням. Цей спосіб дає більш точний результат порівняно з розпізнаванням в реальному часі. Це зумовлено тим що в реальному часі звуковий сигнал містить стороні шуми.

Щоб вибрати аудіо запис який потрібно розпізнати, необхідно натиснути на кнопку «Вибрати файл». Перед користувачем відкриється вікно в якому треба вказати шлях до файлу в операційній системі (Рисунок 5.3).

Важливо щоб аудіо файл, як і при навчанні акустичної моделі, був у форматі MS WAV с частотою дискретизації 16 кГц, з розрядністю 16 біт, з одним каналом. Інші формати система не розпізнає.

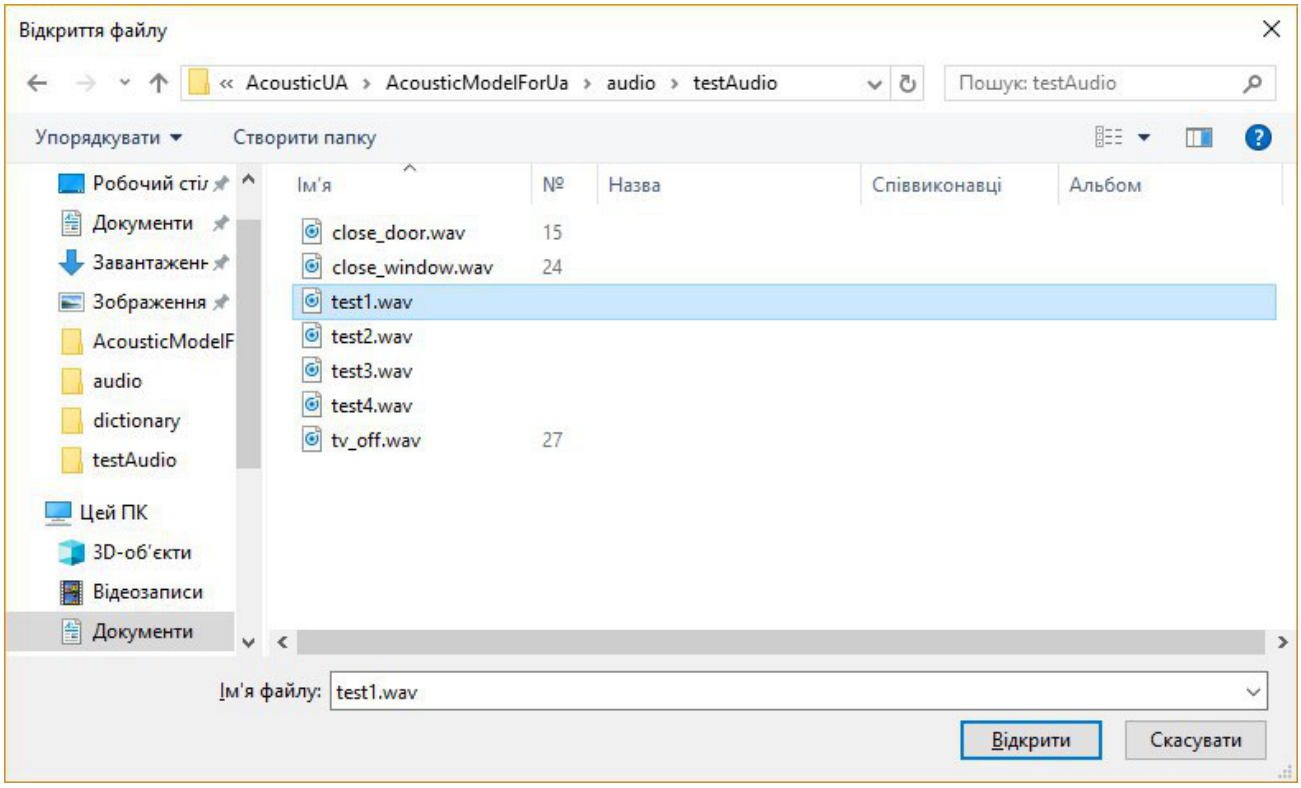

Рисунок 5.3 — Вікно вибору аудіо файлу

Коли файл буде вибрано його ім’я з’явиться біля кнопки «Вибрати файл», це означатиме що файл завантажено і можна провести його розпізнавання (рисунок 5.4).

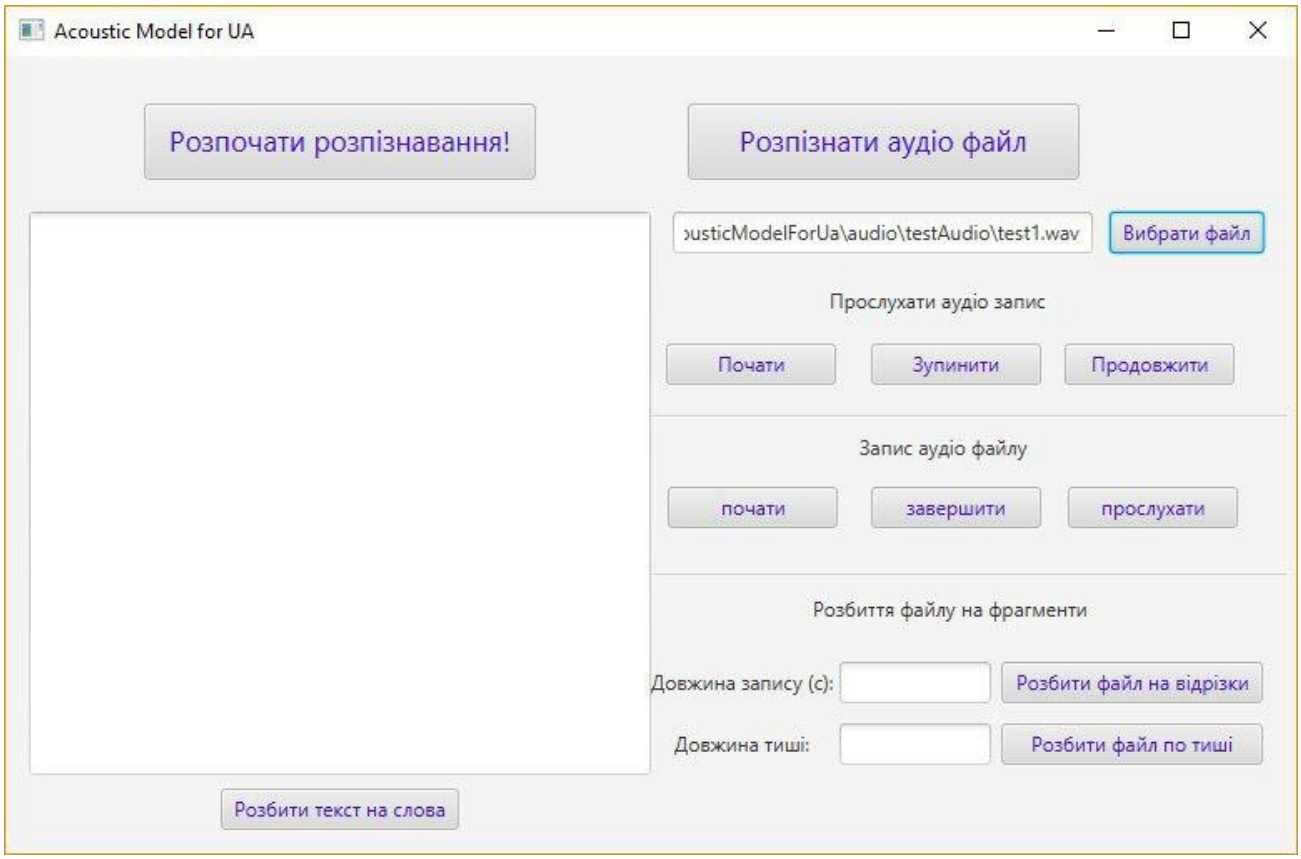

Рисунок 5.4 — Вікно з завантаженим аудіо файлом

Після того як користувач натисне на кнопку «Розпізнати аудіо файл» звуковий потік буде передано до розпізнавача системи, який проаналізує звуковий сигнал. Результат розпізнавання буде відображатись у текстовому полі (рисунок 5.5). Якщо файл буде не правильного формату або пошкоджено, система не зможе його прочитати та видасть помилку.

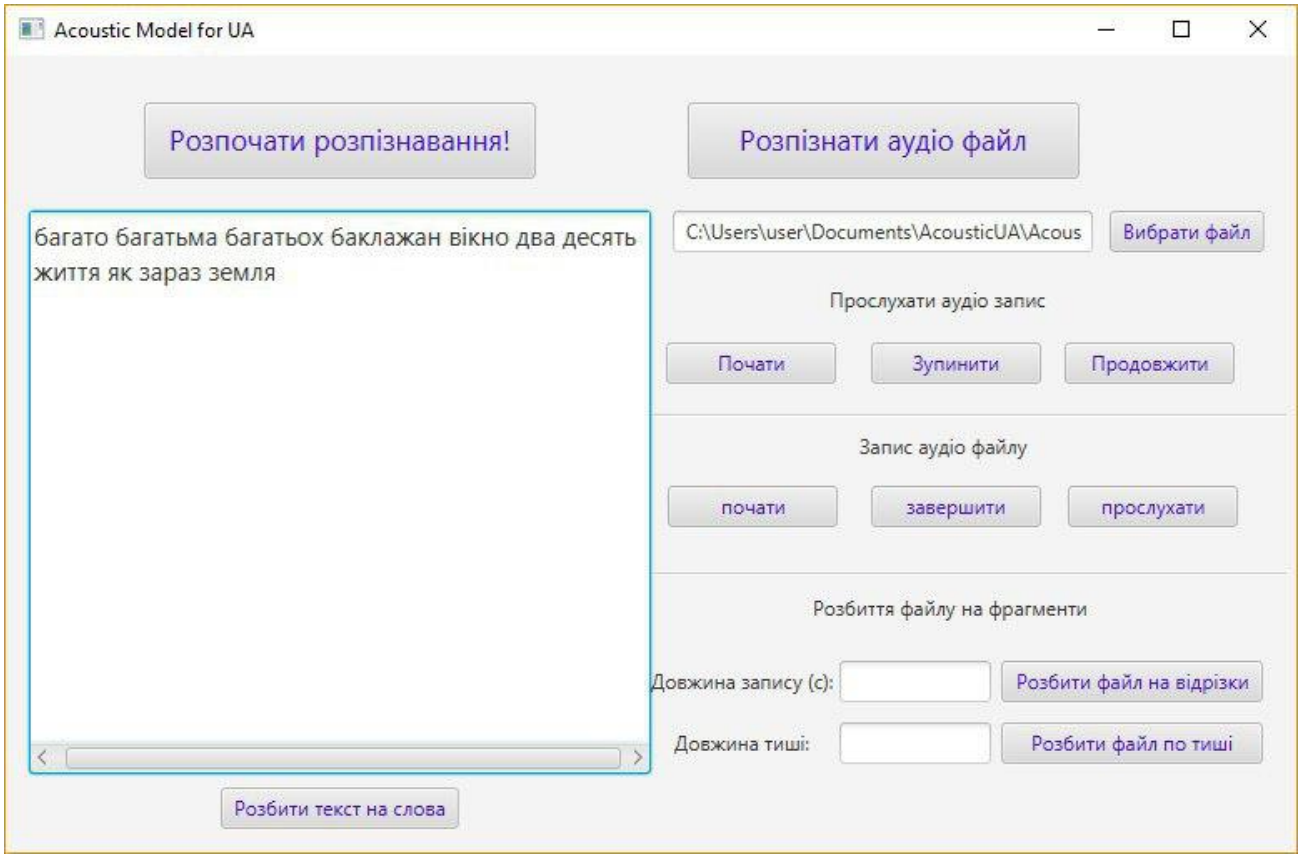

Рисунок 5.5 — Вікно з результатом розпізнавання аудіо файлу

5.3 Інші можливості системи

Окрім основних задач розпізнавання української мови, система має додаткові можливості. Такі як запис звукового файлу та його відтворення, а також розбиття звукового файлу за відрізками тиші або за часовими відрізками.

Щоб записати звуковий файл, треба натиснути на кнопку «почати», після чого почнеться захват звукового сигналу через мікрофон. Щоб завершити запис голосу потрібно натиснути на кнопку «Завершити». Звуковий файл буде збережено автоматично до папки системи (Рисунок. 5.6). Після чого його можна розпізнати.

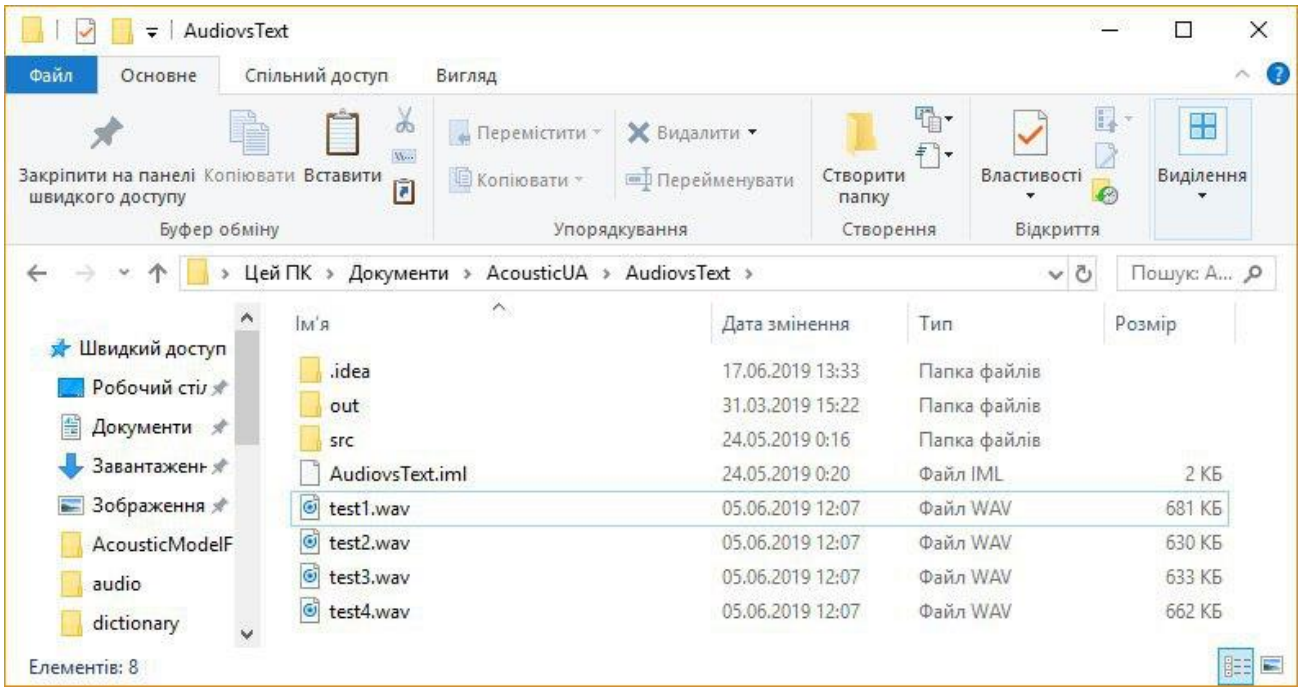

Рисунок 5.6 — Вікно зі створеними аудіо файлами

Щоб розбити звуковий файл за відрізками тиші або за часовим відрізком, треба у відповідному полі вказати час тиші або час звукового файлу після розбиття (Рисунок 5.7).

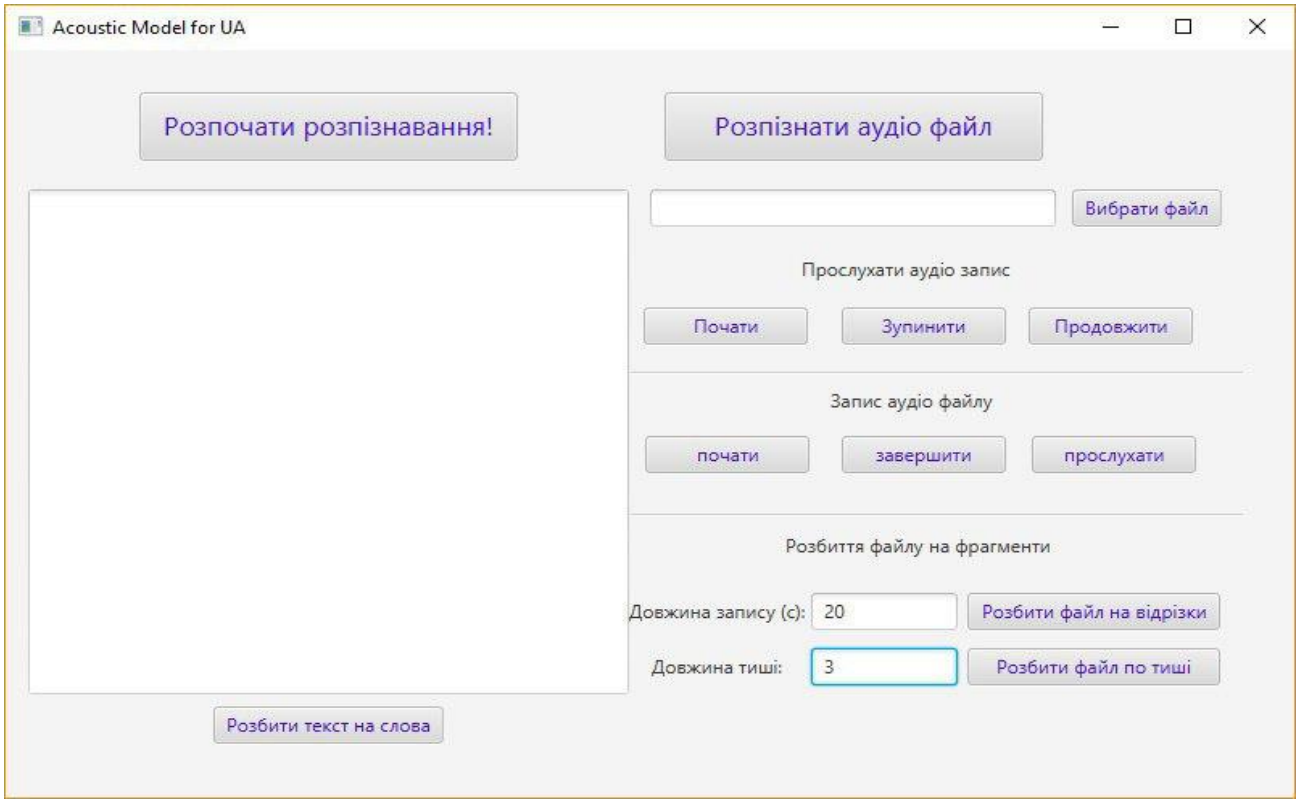

Рисунок 5.7 — Параметри для розбиття файлу.

Тишу система буде виявляти та створювати новий аудіо файл, відзначивши тишу як кінець звукового потоку. Потім можна створити відповідне текстове представлення цих аудіо записів і використовувати їх для навчання акустичної системи.

5.4 Недоліки системи

До недоліків системи можна віднести наступні:

1. Система може розпізнати лише файли в форматі .wav.
2. При розпізнаванні мови в реальному часі, система чутлива до навколишнього середовища. Сторонні шуми можуть знижувати точність розпізнавання.
3. Необхідно розширити базу даних акустичної моделі для підвищення точності розпізнавання.

Висновки до розділу

У даному розділі було розглянуто основні можливості системи та сценарій взаємодії користувача з системою. Також булі розглянуті основні недоліки системи.

ВИСНОВКИ

В ході виконання даної роботи було розроблено акустичну модель для системи розпізнавання української мови.

Додаток було написано на мові програмування Java з використанням графічного інтерфейсу JavaFX.

Програму можна використовувати для систем де потрібен голосовий ввід команд.

Час роботи алгоритму з розпізнавання мови дозволяє його використовувати в режимі реального часу, але обчислювальна потужність мобільних пристроїв не дозволить використовувати алгоритм. Для роботи з даним програмним забезпеченням необхідний комп’ютер досить високої потужності.

В ході роботи було проведено огляд та зроблено аналіз засобів, що були використані для створення даного програмного забезпечення (середовища розробки IntelliJ IDEA, інтерфейсу JavaFx та відповідних бібліотек мови Java).

Для користувача виводяться слова які він вимовив у виді тексту.

Результати роботи програмного продукту, мають досить високу точність розпізнавання. Ця програма робить можливим створення систем управління голосом, що дуже важливо для людей з вадами зору.

У перспективі можливо удосконалення роботи програми за рахунок більш точної настройки мовної моделі або збільшення обсягу словника, але це зажадає великого обсягу даних і часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Reynolds D. Speaker identification and verification using Gaussian mixture speaker models / D. A. Reynolds // Speech Communication. — 1995. — № 17. — С. 91—108.

2. Waheed K. A robust algorithm for detecting speech segments using an entropy contrast : праці міжн. конф. 45th IEEE International Midwest Symposium on Circuits and Systems MWSCAS'2002, 4-7 серп. 2002, Oklahoma (USA). — С. 328—331, III.

3. Building an application with sphinx4 [Електронний ресурс] – Режим доступу до ресурсу: https://cmusphinx.github.io/wiki/tutorialsphinx4/.

4. Г. О. Добрушкін, В. Я. Данилов, Основні підходи до розпізнавання мовної інформації // ANZIP J. – 2000.– Vol. 42, Part E. – P. 96–116.

5. С. Г. Ожаровський, О. А. Мясіщев Метод розпізнання мовлення, по короткий словник, з використання MEL-кепстральних коефіцієнтів // ISSN 2307-5732.

6. Д.А. Гефке, П.М. Зацепин Применение скрытых марковских моделей для распознавания звуковых последовательностей URL: http://izvestia.asu.ru/2012/1-2/info-comp/TheNewsOfASU-2012-1-2-info-comp-03.pdf.

7. Huggins, D. D. An Architecture for Scalable, Universal Speech Recognition / Carnegie Mellon University Pittsburgh, PA, USA ©2011, 44 URL: https://www.lti.cs.cmu.edu/sites/default/files/research/thesis/2011/david_huggins-daines_an_architecture_for_scalable_universal_speech_recognition.pdf

8. И.Б. Тампель, А.А. Карпов АВТОМАТИЧЕСКОЕ РАСПОЗНАВАНИЕ РЕЧИ // Учебное пособие. − СПб: Университет ИТМО, 2016. – 138 с.

9. О.А. Юхименко, В.В. Пилипенко, Р.А. Селюх Адаптація до голосу нового диктора на прикладі спонтанного мовлення з корпусу АКУЕМ // Міжнародний науково-навчальний центр інформаційних технологій та систем.

10. Н.Б. Васильєва, В.В. Пилипенко, О.М. Радуцький, В.В. Робейко, М.М. Сажок. Створення акустичного корпусу українського ефірного мовлення // ТОВ Спеціальні реєструючі системи; м. Київ, Україна – 2009. – № 2. – С. 62–71.

11. Системы распознавания речи с открытым исходным кодом [Електронний ресурс] – Режим доступу до ресурсу: https://lvee.org/en/abstracts/273.

12. Проблемы распознавания речи: что еще предстоит решить [Електронний ресурс] – Режим доступу до ресурсу: https://apptractor.ru/info/articles/problemyi-raspoznavaniya-rechi-chto-eshhe-predstoit-reshit.html.

13. Захват аудио с Java Sound API [Електронний ресурс] – Режим доступу до ресурсу: http://javist.ru/category/javax-sound/.

14. Введение в Java FX [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/javafx/1.1.php.

15. IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/IntelliJ_IDEA.

16. Training an acoustic model for CMUSphinx [Електронний ресурс] – Режим доступу до ресурсу: https://cmusphinx.github.io/wiki/tutorialam/.

17. Дослідження ефективності застосування марковських прихованих моделей для побудови голосових компонент інтерфейсу користувача з програмними додатками [Електронний ресурс] – Режим доступу до ресурсу: https://knowledge.allbest.ru/programming/3c0a65635a2ac79a5d53b88421206d27_1.html

ДОДАТОК А

Акустична модель системи розпізнавання українського мовлення.

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР5276_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР5276_19Б	DBWork.cs Module.cs Form.cs	Основні компоненти
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР5276_19Б	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Акустична модель системи розпізнавання українського мовлення.

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 5

Київ 2019

```
import biz.source_code.dsp.sound.AudioIo;
import edu.cmu.sphinx.api.*;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.FileChooser;

import javax.sound.sampled.*;
import java.awt.*;
import java.io.*;
import java.net.URL;
import java.util.Scanner;
import java.util.regex.Pattern;
public class Recognition {
    @FXML
    public Button beginR;
    @FXML
    public Button beginA;
    @FXML
    public Button Chooser;
    @FXML
    public Button stop;
    @FXML
    public Button start;
    @FXML
    public Button continueBtn;
    @FXML
    public Button startRecording;
    @FXML
    public Button stopRecording;
    @FXML
    public Button captureBtn;
```

```
        @FXML
        public Button splitText;
        @FXML
        public Button splitAudioBySilence;
        @FXML
        public Button splitAudio;
        @FXML
        public TextField fLength;
        @FXML
        public TextField fSilence;
        @FXML
        public TextField textField;
        @FXML
        public TextArea text;
        private Desktop desktop;
        private Sound  sound = null;
        public File file = null;

        Configuration configuration = new Configuration();

        LiveSpeechRecognizer recognizer = null;
        private volatile boolean running = true;
        private volatile boolean paused = false;
        private final Object pauseLock = new Object();

        public static final String INPUT_FILE_LOCATION =
"resources/sampleb.wav";
        public static final int SPLIT_FILE_LENGTH_MS = 500000;

        @FXML
        void Recognition(ActionEvent event) {

        }

        private volatile boolean threadRunning = false;
```

```
        @FXML
        void initialize() {
            desktop = Desktop.getDesktop();
            FileDialog fdlg;

            StringBuilder  sb = new StringBuilder();

configuration.setAcousticModelPath("file:C:\\Users\\user\\Documents\\A
coustic UA\\AcousticModelForUa\\uaadaptcont");

configuration.setDictionaryPath("file:C:\\Users\\user\\Documents\\Acou
sticUA\\AcousticModelForUa\\ua0-out.dict");

configuration.setLanguageModelPath("file:C:\\Users\\user\\Documents\\A
coustic UA\\AcousticModelForUa\\ua.lm.bin");

            beginR.setOnAction( new EventHandler<ActionEvent>() {
                @Override
                public void handle(final ActionEvent e)  {
                    try {
                        recognizer = new
LiveSpeechRecognizer(configuration);
                    } catch (IOException ex) {
                        ex.printStackTrace();
                    }
                    Scanner sc = new Scanner(System.in);
                    recognizer.startRecognition(false);
                    SpeechResult result = recognizer.getResult();
                    OutputStream out = null;
                    String phrase = null;
                    while((result = recognizer.getResult()) != null)
{
                            System.out.format("%s\n",
result.getHypothesis());
                            sb.append(result.getHypothesis());
```

```
                        text.appendText(" " + sb + " ");

                        phrase = sc.nextLine();
                        if (phrase.contains("exit")){
                            recognizer.stopRecognition();
                            break;
                        } else {
                            continue;
                        }



                    }
                    //recognizer.stopRecognition();
                    //text.appendText(String.valueOf(sb));
    }
            });
            beginA.setOnAction( new EventHandler<ActionEvent>() {
                @Override
                public void handle(final ActionEvent e)  {
                    String nameFile = file.getAbsolutePath();
                    StreamSpeechRecognizer recognizer = null;
                    try {
                        recognizer = new
StreamSpeechRecognizer(configuration);
                    } catch (IOException ex) {
                        ex.printStackTrace();
                    }
                    InputStream stream = null;
                    try {
                        stream = new FileInputStream(new
File(nameFile));
                    } catch (FileNotFoundException ex) {
                        ex.printStackTrace();
                    }
```

```
                    recognizer.startRecognition(stream);
                    SpeechResult result;
                    while ((result = recognizer.getResult()) != null)
{
                        System.out.format("%s\n",
result.getHypothesis());
                        sb.append(result.getHypothesis());
                        text.appendText(" " + sb + " ");
                    }
                    recognizer.stopRecognition();

            }
        });
     Chooser.setOnAction( new EventHandler<ActionEvent>() {
            @Override
            public void handle(final ActionEvent e) {

                FileChooser fileChooser = new FileChooser();
                textField.clear();
                file = fileChooser.showOpenDialog(null);
                if (file != null) {
                    textField.appendText(file.getAbsolutePath() +
"\n");
                }
                sound = new Sound(file.getAbsoluteFile());
                //startThread(file.getAbsoluteFile());
            }
        });
            start.setOnAction( new EventHandler<ActionEvent>() {
            @Override
            public void handle(final ActionEvent e) {

                t.start();
```

```
            }
        });
            stop.setOnAction( new EventHandler<ActionEvent>() {
            @Override

            public void handle(final ActionEvent e) {

            }
        });
            continueBtn.setOnAction( new EventHandler<ActionEvent>()
{
            @Override
            public void handle(final ActionEvent e) {

                t.notifyAll();

            }
        });

            splitText.setOnAction( new EventHandler<ActionEvent>() {
            @Override
            public void handle(final ActionEvent e) {

                if(!text.getText().isEmpty()) {
                    String string = text.getText();
                    String[] parts = string.split(Pattern.quote(".!?
"));
                    for (int i=0; i<parts.length;i++) {
                        text.setText("Successful");
                        saveToFile(parts[i],i);
                    }
                }
            }
        });
    }
```

```
     static void saveToFile(String text,int i) {
             PrintWriter out = null;
             try {
                 out = new PrintWriter(new
FileOutputStream("text"+i+".txt"));
                 out.print(text);
             } catch (FileNotFoundException e) {
                 e.printStackTrace();
             } catch (IOException e) {
                 e.printStackTrace();
             } finally {
                 if (out != null) {
                     out.flush();
                     out.close();
                 }
             }
         }

         void saveToWave( ByteArrayOutputStream
                                 byteArrayOutputStream) throws
IOException{
             File out = new File("SaveYourVoice.wav");
             AudioFormat format;
             format = getAudioFormat();
             byte[] recordData = byteArrayOutputStream.toByteArray();
             ByteArrayInputStream bais = new
ByteArrayInputStream(recordData);
             AudioInputStream audioInputStream;
             audioInputStream = new AudioInputStream(bais,
format,recordData.length/ format.getFrameSize());
             AudioSystem.write(audioInputStream,
AudioFileFormat.Type.WAVE, out);
             audioInputStream.close();
         }
```

ДОДАТОК В

Акустична модель системи розпізнавання українського мовлення.

Опис програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 8

Київ 2019

АНОТАЦІЯ

Розділ містить опис системи розпізнавання української мови в реальному часі або в звуковому файлі. Створений програмний продукт використовує українську акустичну модель, мовну модель та фонетичний словник для української мови. Щоб отримати розпізнаний текст з голосового сигналу система виконує такі завдання:

– зчитування звукового потоку отриманого з мікрофону;

– аналіз звукового сигналу на основі мовної моделі;

– вивід отриманого тексту на екран;

Дану програму було написано на платформі Java Machine, використовуючи бібліотеку Sphinx4, мову Java та Java FX.

Зміст

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається опис системи розпізнавання української мови. У додатку Б знаходиться код реалізації основних методів системи.

Систему реалізовано за допомогою мови програмування Java та за допомогою інтсрумента Java FX для створення графічного інтерфейсу. Щоб скористатися розробленим додатком необхідно встановити середовище розробки IntelliJ IDEA, Java Machine, та налаштувати мікрофон або записати аудіофайл у форматі .wav.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням системи розпізнавання це розпізнавання українського мовлення у звуковому сигналі. Систему можна використовувати для розпізнавання як в реальному часі так і розпізнавання звукового файлу.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Розпізнавати українську мову э головним завданням системи. При запуску системи, з мікрофону починається зчитування звукового сигналу. Система аналізує цей сигнал та показує результат розпізнавання в текстовому полі.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Систему розроблено у середовищі розробки IntelliJ IDEA, що забезпечує набір сервісних функцій та графічний діалог з користувачем, на комп’ютері, що використовував операційну систему Windows 10. Для побудови графічного інтерфейсу використовувався набір інструментів Java FX.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний продукт реалізований як окремий клас, який забезпечує існування клієнтської частини та бізнес-логіки окремо від одного, але разом із цим запуск обох компонентів відбувається одночасно.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є звуковий запис, який потрібно розпізнати.

Вихідними даними програмного модуля є текстова інформація, яка міститься у звукозаписі.