ДОДАТОК А

Лістинг файлу «voice_recognition_1.py»

```
import queue
import sounddevice as sd # Access audio data from microphone
import vosk              # Offline speech recognition
import json
import words             # keywords
from skills import *     # skills import
from sklearn.feature_extraction.text import CountVectorizer
                         # Machine Learning
from sklearn.linear_model import LogisticRegression
                         # Machine Learning

q = queue.Queue()
model = vosk.Model('model_small')
device = sd.default.device = 3, 1
                         # sd.default.device = input, output
samplerate         =         int(sd.query_devices(device[1],
'input')['default_samplerate'])

def callback(indata, frames, time, status):
    q.put(bytes(indata))

def recognize(data, vectorizer, clf):
                         # Recognition of calls to a bot
    trg = words.TRIGGERS.intersection(data.split())
    if not trg:
        return

    data = data.replace(list(trg)[0], '').strip()
    text_vector = vectorizer.transform([data]).toarray()[0]
    answer = clf.predict([text_vector])[0]

    func_name = answer.split()[0]
    speaker(answer.replace(func_name, ''))# Voice acting
    exec(func_name + '()')# Decoding the string

def main():
    vectorizer = CountVectorizer()
                                        vectors                =
vectorizer.fit_transform(list(words.data_set.keys()))  #  Getting
a dictionary and keys, hashing them, converting them to a list,
passing them to a method

    clf = LogisticRegression()
    clf.fit(vectors, list(words.data_set.values()))
# Matching vectors and responses
    del words.data_set
```

```
        #blocksize - The amount of information sent for processing
at a time
         with sd.RawInputStream(samplerate=samplerate, blocksize
=    16000,    device=device[1],    dtype='int16',    channels=1,
callback=callback):

            rec = vosk.KaldiRecognizer(model, samplerate)
            print('Listening')
            while True:
                data = q.get()
                if rec.AcceptWaveform(data):
                    data = json.loads(rec.Result())['text']
                    recognize(data, vectorizer, clf)
        #   If I need to see temporary result:
                    print(rec.Result())
                else:
                    print(rec.PartialResult())

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

ДОДАТОК Б

Лістинг файлу «app.py»

```
import tkinter
import tkinter.messagebox
import customtkinter
import os, subprocess

customtkinter.set_appearance_mode("System")  # Modes: "System" (standard), "Dark", "Light"
customtkinter.set_default_color_theme("blue")  # Themes: "blue" (standard), "green", "dark-blue"

class App(customtkinter.CTk):
    def __init__(self):
        super().__init__()

        # configure window
        self.title("Voice Recognition")
        self.geometry(f"{1100}x{580}")

        # configure grid layout (4x4)
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure((2, 3), weight=0)
        self.grid_rowconfigure((0, 1, 2), weight=1)

        # create sidebar frame with widgets
        self.sidebar_frame = customtkinter.CTkFrame(self, width=140, corner_radius=0)
        self.sidebar_frame.grid(row=0, column=0, rowspan=4, sticky="nsew")
        self.sidebar_frame.grid_rowconfigure(4, weight=1)
        self.logo_label = customtkinter.CTkLabel(self.sidebar_frame, text="Voice Recognition", font=customtkinter.CTkFont(size=20, weight="bold"))
        self.logo_label.grid(row=0, column=0, padx=20, pady=(20, 10))
        self.button_listen = customtkinter.CTkButton(self.sidebar_frame, text="Listen", command=self.button_listen)
        self.button_listen.grid(row=1, column=0, padx=20, pady=10)
        self.button_new_command = customtkinter.CTkButton(self.sidebar_frame, text="New command", command=self.button_new_command)
        self.button_new_command.grid(row=2, column=0, padx=20, pady=10)
        self.button_delete_command = customtkinter.CTkButton(self.sidebar_frame, text="Delete
```

```
command", command=self.button_delete_command)
                self.button_delete_command.grid(row=3,  column=0,
padx=20, pady=10)
                                    self.appearance_mode_label      =
customtkinter.CTkLabel(self.sidebar_frame,       text="Appearance
Mode:", anchor="w")
                self.appearance_mode_label.grid(row=5,  column=0,
padx=20, pady=(10, 0))
                                self.appearance_mode_optionemenu     =
customtkinter.CTkOptionMenu(self.sidebar_frame,  values=["Light",
"Dark", "System"],

           command=self.change_appearance_mode_event)
                    self.appearance_mode_optionemenu.grid(row=6,
column=0, padx=20, pady=(10, 10))
                                            self.scaling_label       =
customtkinter.CTkLabel(self.sidebar_frame,   text="UI   Scaling:",
anchor="w")
                self.scaling_label.grid(row=7,  column=0,  padx=20,
pady=(10, 0))
                                    self.scaling_optionemenu      =
customtkinter.CTkOptionMenu(self.sidebar_frame,    values=["80%",
"90%", "100%", "110%", "120%"],

   command=self.change_scaling_event)
                self.scaling_optionemenu.grid(row=8,  column=0,
padx=20, pady=(10, 20))

            # create textbox
                self.textbox  =  customtkinter.CTkTextbox(self,
width=250)
                self.textbox.grid(row=0,  column=1,  padx=(20,  0),
pady=(20, 0), sticky="nsew")

            # create tabview
                self.tabview  =  customtkinter.CTkTabview(self,
width=250)
                self.tabview.grid(row=0,  column=2,  padx=(20,  0),
pady=(20, 0), sticky="nsew")
            self.tabview.add("Guest book")
            self.tabview.add("Book of complaints")
                                            self.tabview.tab("Guest
book").grid_columnconfigure(0,  weight=1)   # configure grid of
individual tabs
                                    self.tabview.tab("Book      of
complaints").grid_columnconfigure(0, weight=1)

                                            self.label_tab_1        =
customtkinter.CTkLabel(self.tabview.tab("Guest           book"),
text="Wellcome to the Guest book!\n" + "Here you can leave your
wishes and share pleasant impressions.\n" + "If you would like to
report a malfunction or service interruption, please,\n" +
"proceed to the Book of complaint.\n")
```

```
                self.label_tab_1.grid(row=0,  column=0,  padx=20,
pady=20)
                                self.string_input_button_guest     =
customtkinter.CTkButton(self.tabview.tab("Guest           book"),
text="Open the Guest Book",
                                                                   c
ommand=self.open_input_dialog_event)
                self.string_input_button_guest.grid(row=2, column=0,
padx=20, pady=(10, 10))
                                            self.label_tab_2       =
customtkinter.CTkLabel(self.tabview.tab("Book  of  complaints"),
text="Wellcome to the Book of complaints!\n" + "Here you can report
a malfunction or service interruption.\n" + "If you would like to
leave your wishes and share your pleasant impressions,\n" + "please
proceed to the guest book.\n")
                self.label_tab_2.grid(row=0,  column=0,  padx=20,
pady=20)
                            self.string_input_button_complaint    =
customtkinter.CTkButton(self.tabview.tab("Book  of  complaints"),
text="Open the Book of complaints",
                                                                   c
ommand=self.open_input_dialog_event)
                    self.string_input_button_complaint.grid(row=2,
column=0, padx=20, pady=(10, 10))

            # create slider and progressbar frame
                                self.slider_progressbar_frame     =
customtkinter.CTkFrame(self, fg_color="transparent")
                self.slider_progressbar_frame.grid(row=1, column=1,
padx=(20, 0), pady=(20, 0), sticky="nsew")
            self.slider_progressbar_frame.grid_columnconfigure(0,
weight=1)
                self.slider_progressbar_frame.grid_rowconfigure(4,
weight=1)

                                            self.progressbar_1       =
customtkinter.CTkProgressBar(self.slider_progressbar_frame)
                self.progressbar_1.grid(row=1, column=0, padx=(10,
10), pady=(10, 10), sticky="ew")

            self.appearance_mode_optionemenu.set("Dark")
            self.scaling_optionemenu.set("100%")
            self.progressbar_1.configure(mode="indeterminnate")
            self.progressbar_1.start()
               self.textbox.insert("0.0",  "Hello,  User!\n\n"  +
"Hello, I am your voice assistant. I have an intuitive interface.
If you want me to start working, click the 'Listen' button.\n\n")

        def open_input_dialog_event(self):
            dialog = customtkinter.CTkInputDialog(text="Type your
review:", title="Review section")
            print("Review:", dialog.get_input())
```

```
                    def change_appearance_mode_event(self,
new_appearance_mode: str):
            customtkinter.set_appearance_mode(new_appearance_mod
e)

        def change_scaling_event(self, new_scaling: str):
            new_scaling_float = int(new_scaling.replace("%", ""))
/ 100
            customtkinter.set_widget_scaling(new_scaling_float)

        def sidebar_button_event(self):
            print("sidebar_button click")


        def button_listen(self):
            os.startfile('voice_recognition.py')
            #subprocess.Popen('voice_recognition.py')
            print("button_listen click")

        def button_new_command(self):
            dialog = customtkinter.CTkInputDialog(text="Type your
command:", title="New command")
            print("New command:", dialog.get_input())
            #temp_command = dialog.get_input()


        def button_delete_command(self):
            print()

    if __name__ == "__main__":
        app = App()
        app.mainloop()
```

ДОДАТОК В

Лістинг файлу «skills.py»

```
    import os, webbrowser, sys, requests, subprocess, pyttsx3

    engine = pyttsx3.init()                          # initialization
speach engine
    engine.setProperty('rate', 180)                 # speech speed
    engine.setProperty('voice', 'English')          # speech language

    def speaker(text):
        engine.say(text)
        engine.runAndWait()

    #Base
    def offBot():
        sys.exit()
    def passive():
        pass

    #Lights
    def lightson_lobby():
        #Lights script
        print('Lobby lights on')
    def lightsoff_lobby():
        #Lights script
        print('Lobby lights off')
    def lightson_Mianroom():
        #Lights script
        print('Main room lights on')
    def lightsoff_Mainroom():
        #Lights script
        print('Main room lights off')
    def lightson_Bedroom():
        #Lights script
        print('Bedroom lights on')
    def lightsoff_Bedroom():
        #Lights script
        print('Bedroom lights off')

    #Smart home
    def water_heat():
        #Water heating script
        print('Heating')

    #Speaking
    def speak():
        print('Still breathing')
        #Speaking script
```

```
        #Speaking script
    def weather():
        #Weather script
        print('hello')

    #Medicine
    def pills():
        #Pills script
        #if time is ok
        print('Yes, it`s time to take pills')

    #Software
    def browser():
        webbrowser.open('https://www.google.com', new=2)
    def game():
         subprocess.Popen('D:\Games\Games\The Binding of Isaac
Rebirth\isaac-ng.exe')
```

ДОДАТОК Г

Лістинг файлу «words.py»

```
TRIGGERS = {'бот', 'буд', 'бут', 'дім', 'будинок', 'команда'}


#382693 бот big
#128762 бот small
data_set = {

#Base
'виключай систему': 'offBot Turning off',
'виключає систему': 'offBot Turning off',
'виключаю систему': 'offBot Turning off',

#Smart home
'відчини двері': 'passive Opening doors',
'відчинить двері': 'passive Opening doors',
'відчинять двері': 'passive Opening doors',
'відчиняй двері': 'passive Opening doors',
'відчиню двері': 'passive Opening doors',

'виключи воду': 'passive Turning off the taps',
'виключай воду': 'passive Turning off the taps',
'виключу воду': 'passive Turning off the taps',
'виключиш воду': 'passive Turning off the taps',
'закрий воду': 'passive Turning off the taps',
'закривай воду': 'passive Turning off the taps',
'закриваю воду': 'passive Turning off the taps',
'перекрий воду': 'passive Turning off the taps',

'включи воду': 'passive Turning on the taps',
'включай воду': 'passive Turning on the taps',
'включу воду': 'passive Turning on the taps',
'включиш воду': 'passive Turning on the taps',
'відкрий воду': 'passive Turning on the taps',
'відкривай воду': 'passive Turning on the taps',
'відкрию воду': 'passive Turning on the taps',

'включи вентиляцію': 'passive Turning on the ventilation',
'включає вентиляцію': 'passive Turning on the ventilation',
'включив вентиляцію': 'passive Turning on the ventilation',
'включить вентиляцію': 'passive Turning on the ventilation',
'включи вентиляцією': 'passive Turning on the ventilation',

'включи світло в прихожій':'lightson_lobby Lobby lights on',
'включай світло в прихожій':'lightson_lobby Lobby lights on',
'включити світло в прихожій':'lightson_lobby Lobby lights
on',
```

```
    'включив світло в прихожій':'lightson_lobby Lobby lights on',
    'включить світло в прихожій':'lightson_lobby Lobby lights
on',

    'виключи світло в прихожій':'lightsoff_lobby Lobby lights
off',

    'включи світло в кімнаті':'lightson_Mainroom Main room lights
on',

    'виключи світло в кімнаті':'lightsoff_Mainroom Main room
lights off',

    'включи світло в спальні':'lightson_Bedroom Bedroom lights
on',

    'виключи світло в кімнаті':'lightsoff_Bedroom Bedroom lights
off',

    'підігрій воду':'water_heat Heating the water',


    #Speaking
    'як погода': 'weather Displaying weather information:',
    'не пора мені приймати пігулки':'pills Checking',
    'як справи': 'speak',

    #Software
    'браузер': 'browser Browser opening',
    'хочу пограти в гру': 'game Launching the game...',


    }
```

ДОДАТОК Ґ

Лістинг файлу «voice_recognition_2.py»

```
import speech_recognition as sr # (Speech-To-Text)
import os
import sys
import webbrowser

def command():
    print("Listening")
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.pause_threshold = 1
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source)
    try:
        mission = r.recognize_google(audio).lower()
        print("you just said: "+ mission)
    except sr.UnknownValueError:
        mission = command()
    return mission

def makeMission(mission):
    print("Listening")
    if "university" in mission:
        print("Opening your university website")
        url = "https://dl.nure.ua/"
        webbrowser.open(url)
    elif "hello" in mission:
        print("Hi, can I help you?")
    elif "stop" in mission:
        print("Closing program")
        sys.exit()
while True:
    makeMission(command())
```

ДОДАТОК Д

Графічний матеріал кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра «Електронних обчислювальних машин»

Презентація до кваліфікаційної роботи на тему:
«Методи розпізнавання голосу для керування системою розумний будинок»

Виконав:
ст. гр. СПм-22-6
Міхайлов Ілля Олегович

Харків 2024

1

Мета та обґрунтування

Метою роботи є аналіз методів розпізнавання голосу, наочне порівняння, визначення найбільш підходящих для розумного будинку та розробка рекомендацій щодо їх реалізації. Очікується, що результати будуть цінними як для теоретиків, так і для практиків.

Розумний будинок – це комплекс рішень для автоматизації повсякденних дій. В розумному будинку може бути реалізовано управління голосом для більш зручного для користувача способу керування будинком.

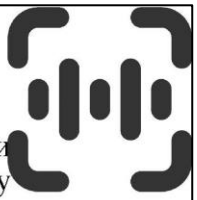
Голосове керування дозволяє покращити рівень автоматизації в домашньому середовищі, роблячи його більш інтелектуальним та ефективним. Голосові команди можуть бути пов'язані з різними сценаріями автоматизації, такими як керування освітленням, опаленням, кондиціюванням повітря, аудіо-відео системами та безпекою.

2

Існуючі рішення

Назва	Зображення	Плюси	Мінуси
Google Nest		Інтеграція з екосистемою Google: Google Assistant, Google Home та ін. Простота використання Гідна енергоефективність	Залежність від Інтернету Велика вартість Погана сумісність з девайсами інших виробників
Apple HomeKit		Інтеграція з екосистемою Apple: Siri, Apple Home та ін. Приватність та безпека даних Apple регулярно надає оновлення програмного забезпечення	Обмежений вибір пристроїв для розумного будинку Велика вартість Залежність від Інтернету
Amazon Echo		За допомогою пристроїв Echo можна створювати розумні рутини, автоматизуючи завдання та сценарії. Гарна інтеграція зі сторонніми онлайн сервісами Широкий вибір пристроїв	Приватність та дані: пов'язано зі збором аналітичних даних, що може викликати недовіру у користувача Залежність від Інтернету Можливі проблеми із розпізнаванням голосових команд.

Дослідження методів розпізнавання голосу



Для того, щоб реалізувати голосове управління потрібно розуміти алгоритм роботи розпізнавання голосу. Підхід до побудування алгоритму розпізнавання голосу також називається методом розпізнавання.

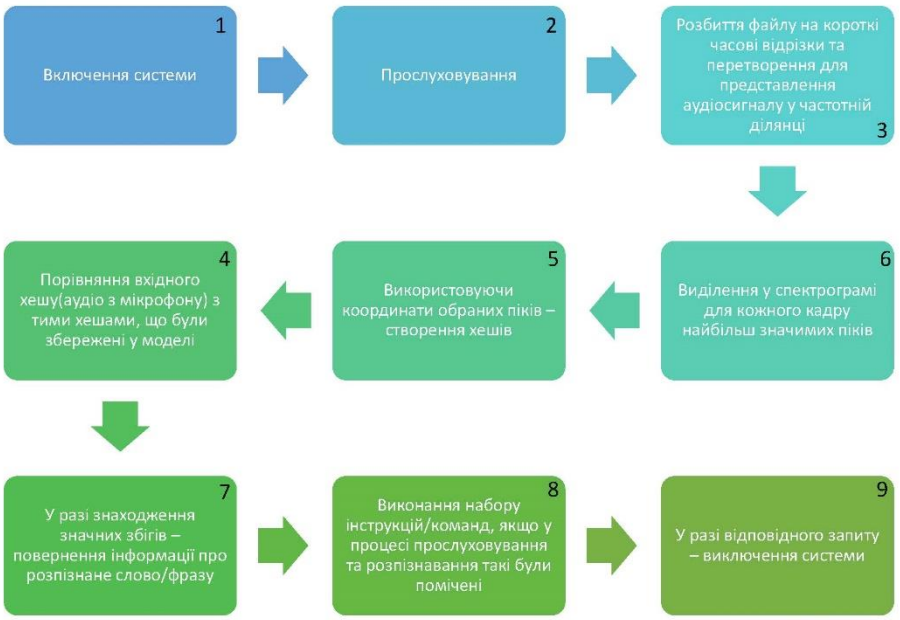
У кваліфікаційній роботі було розглянуто два таких методи:

- Offline розпізнавання голосу, реалізоване за допомогою хешування та бібліотеки vosk;

- Online метод розпізнавання голосу шляхом перетворення аудіодоріжки на масив з частотою дискретизації.

Ці методи будуть обґрунтовані та розібрані на наступних слайдах.

Дослідження offline методу розпізнавання голосу, реалізоване за допомогою хешування та бібліотеки vosk



5

Дослідження online методу розпізнавання голосу, реалізоване шляхом перетворення аудіодоріжки на масив з частотою дискретизації

6

Програмна реалізація застосунку

Через те, що розуміння алгоритму роботи методу не надає змогу користуватися їм, було вирішено програмно реалізувати ці методи мовою Python, а відладка була виконана у середовищі Visual Studio Code.

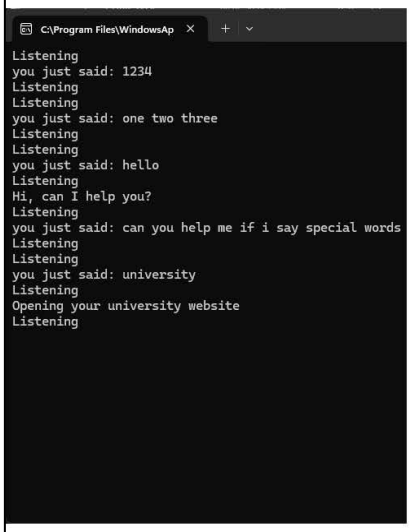
Використання offline методу розпізнавання голосу було реалізовано у відповідності до алгоритму роботи методу хешування – при запуску застосунку починається прослуховування прямої мови користувач.

А на виході застосунок ефективно виконує набір команд, що було сформовано для тієї чи іншої команди, промовленої користувачем.

Паралельно з цим виконуються функції, що задіюють процеси розбиття аудіодоріжки, перетворення спектрограми на хеш, порівняння хешу та ін.

7

Програмна реалізація застосунку



Реалізація online методу розпізнавання голосу також була зроблена у відповідності до розглянутого алгоритму:

включення системи;

початок прослуховування;

при паузі мовлення – перетворення аудіодоріжки на цифровий формат(масив);

обробка аудіо даних;

передача вхідних даних до моделі;

повернення інформації щодо розпізнаних слів/фраз;

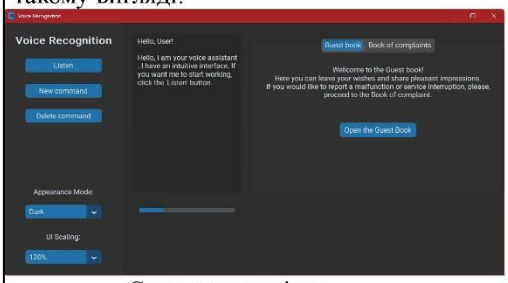
у разі виявлення команди – її виконання.

Реалізовані застосунки повністю охоплюють спектр можливостей методів, що розглядаються. Застосунки розпізнають голос використовуючи той чи інший метод та видають інформацію про розпізнаний текст. Для зручності користувача було вирішено розробити інтуїтивно зрозумілий інтерфейс.

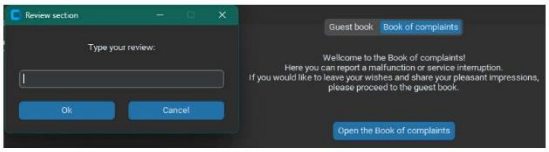
8

Реалізація зручного інтерфейсу

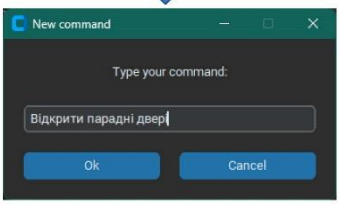
Після дослідження алгоритмів роботи методів розпізнавання голосу їх було програмно реалізовано та додано інтерфейс для зручності майбутнього користувача. Користувальницький досвід можна представити в такому вигляді:



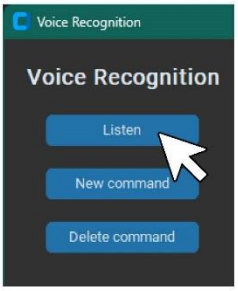
Стартова сторінка



Написання скарги/побажання



Додання/видалення команди



Активація розпізнавання голосу

9

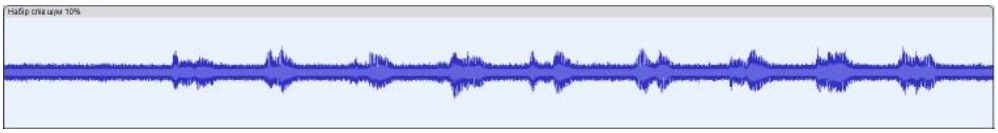
Експериментальне дослідження з метою порівняння методів

Для відповідності меті кваліфікаційної роботи потрібно порівняти реалізовані методи та надати рекомендації щодо використання їх у системі розумний будинок.

Зважаючи на те, що в кожного окремого користувача різна якість вхідного пристрою(мікрофону) – доцільніше було б порівнювати один аудіозапис, але з різною кількістю білого монотонного шуму на ньому. Саме тому було вирішено реалізувати підхід до порівняння методів таким чином:

- зробити запис 10 коротких слів (нижче відображено волноформу цього запису);

- на цей запис накласти різну кількість шуму від 1 до 10% (нижче відображено волноформу того ж запису, але з шумом 10%;



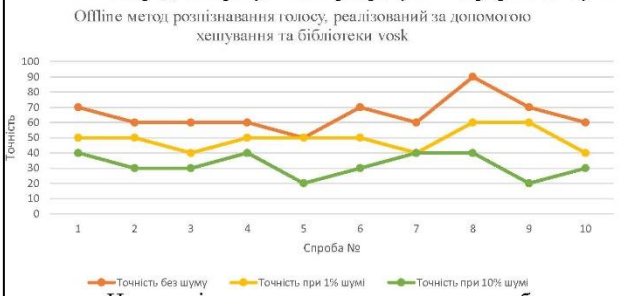
- після чого подати кожну запис на вхід застосунку, що використовує той чи інший метод та дослідити точність, затримку та надійність кожного з методів під час розпізнавання різних аудиодоріжок .

10

Експериментальне дослідження з метою порівняння методів

За наявності підготованих аудіозаписів проводяться тестові випробування. Точність розраховувалася за такою формулою: точність = Accuracy $= \frac{W}{C} * 100\%$, де W – кількість правильно розпізнаних слів, C – загальна кількість слів.

Усереднені результати розрахунків сформовані у вигляді графіків:



На основі отриманих даних можливо зробити висновок щодо того, що точність Online методу є набагато вищою, а саме на 15,7% у середньому. Знаючи точність методів при роботі з кожною аудіодоріжкою можливо зробити розрахунок надійності кожного методу. Надійність розраховувалася за такою формулою: надійність = Reliability $= \frac{A_1+A_2+A_3}{3}$, де A_1 – точність при відсутності шуму, A_2 – точність при 1% шумі, A_3 – точність при 10% шумі. Результати було представлено у вигляді таблиці:

	1	2
Точність без шуму	65%	80%
Точність при 1% шумі	49%	66%
Точність при 10% шумі	32%	47%
Надійність	48.6%	64.3%

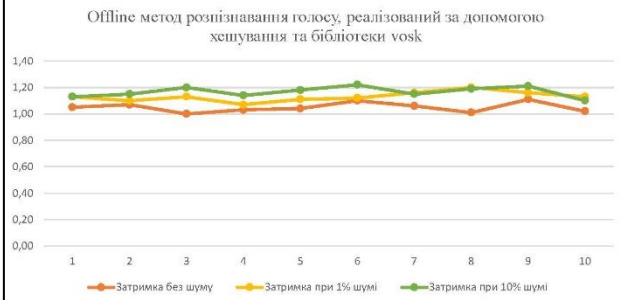
Експериментальне дослідження з метою порівняння методів

Наступне тестування буде проводитись з метою дослідження швидкості роботи методу.

Швидкість роботи методу розпізнавання голосу визначалася як затримка від початку до кінця промови користувача. Це може бути як велике речення, так і одне слово. Для цього дослідження використовувалися ті ж аудіодоріжки, що й при дослідженні точності методу. Затримка рахувалася за такою формулою:

затримка = Delay $= \frac{T}{C}$, де T – час розпізнавання всіх слів C – загальна кількість слів.

Усереднені результати розрахунків сформовані у вигляді графіків:



Як можна побачити з результатів – не дивлячись на перевагу у точності та надійності, Online метод сильно програє у швидкості. А саме на 38% нижче була затримка у Offline методу.

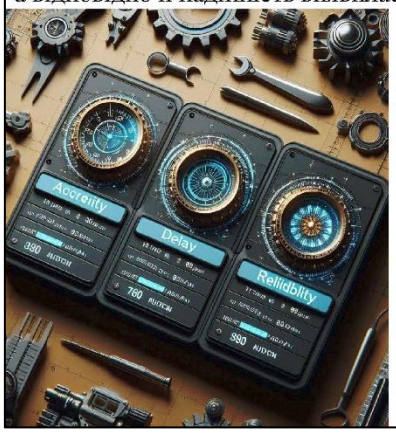
Отримані результати дають можливість під ще одним кутом подивитись на порівняння двох різних методів. Детальніше це буде розписано на наступному слайді.

Висновки

Фінальні результати експериментального дослідження можна відобразити у вигляді таблиці:

		1	2
Без шуму	Точність	65%	80%
	Затримка	1.04	1.46
Шум 1%	Точність	49%	66%
	Затримка	1.13	1.51
Шум 10%	Точність	32%	47%
	Затримка	1.16	1.64
Надійність		**48.6%**	**64.3%**

Як можна побачити з таблиці показники для різних методів дуже різняться. Точність у методу, що заснований на Offline розпізнаванні з хешуванням є нижчою, ніж у Online методу з перетворенням на масив, а саме на 15,7 % точність, а відповідно й надійність виявилася нижче.



Натомість такий показник як затримка у Offline методі виявився більш позитивним. На цілих 38% у середньому кращій результат виявився у цього методу.

Маючи такі результати порівняння методів, можливо порекомендувати саме Online метод з перетворенням на масив для керуванням системою «Розумний будинок» через вищу точність та надійність, а також можливості, що надає підключення до глобальної мережі. Загальні рекомендації щодо використання того чи іншого методу представлені наступним чином:

- Offline метод:

1. розпізнавання голосу в автономних пристроях (розумний годинник, портативна акустика)
2. розпізнавання голосу в умовах обмеженого доступу до інтернету
3. де потрібна більша швидкість, ніж точність

- Online метод:

1. Пристрої, що отримують багато переваг, маючи доступ до інтернету(хаб розумного будинку, смартфон)
2. розпізнавання мови у складних акустичних умовах

13

NURE

Харківський національний університет радіоелектроніки

Стаття «Забезпечення заощадливого ресурсоспоживання житловим простором через контроль мікроклімату розумного будинку»

DOI: https://doi.org/10.32999/ksu2307-8030/2023-49-1

Була викладена у 49 випуску «Наукового вісника Херсонського державного університету. Серія «економічні науки»

Дякую за увагу

Виконав ст. гр. СПм-22-6

Міхайлов І. О.

Харків 2024

14