Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Програмний модуль голосового асистента для синтезу та розпізнавання мовлення
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Кулик А. О.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник д.т.н., проф. каф. БМІ Сніжко Д. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В. О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кулику Антону Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмний модуль голосового асистента для синтезу та розпізнавання мовлення

затверджена наказом університету від 31 березня 20 23 р. № 306Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 2023 р.
3. Вихідні дані до роботи: Науково-технічні публікації, дані Інтернет-джерел, відомі наукові проекти та статті про обрану предметну область, документація використання обраних технологій для розробки власної програми, методичні вказівки до виконання кваліфікаційної роботи магістра

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної області
2) Огляд існуючих віртуальних асистентів
3) Виділення основних недоліків сучасних голосових асистентів
4) Аналіз методів для створення персонального голосового помічника
5) Огляд архітектурних властивостей голосового асистента
6) Постановка та опис структурних та функціональних особливостей задачі
7) Вибір необхідних технологій для програмної реалізації персонального голосового асистента
8) Розробка ключових модулів та компонентів голосового помічника
9) Проведення аналізу та опису функціональних можливостей створеної прикладної програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – Базова архітектура віртуального голосового асистента, Рисунок 2 – Файлова архітектура проекту розробленого голосового асистента, Рисунок 3 – Діаграма класів розробленої прикладної програми віртуального голосового асистента, Рисунок 4 – Архітектура моделі Squeezeformer-CTC, Рисунок 5 – Схема методики навчання моделі VITS, Рисунок 6 – Діаграма прецедентів використання голосового асистента

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної галузі | 05.04.2023 | виконано |
| 2 | Аналіз методів для створення персонального | 11.04.2023 | виконано |
| | голосового асистента | | |
| 3 | Огляд архітектурних властивостей голосових | 15.04.2023 | виконано |
| | помічників | | |
| 4 | Опис структурних та функціональних особливостей | 20.04.2023 | виконано |
| | поставленої задачі | | |
| 5 | Програмна реалізація власного голосового | 25.04.2023 | виконано |
| | асистента | | |
| 6 | Написання пояснювальної записки | 10.05.2023 | виконано |
| 7 | Попередній захист | 15.05.2023 | виконано |
| 8 | Захист перед ЕК | 19.05.2023 | |

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ д.т.н., проф. каф. БМІ Сніжко Д. В.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 62 с., 1 табл., 17 рис., 1 дод., 11 джерел.

ГОЛОСОВИЙ АСИСТЕНТ, ГОЛОСОВИЙ ІНТЕРФЕЙС, ОБРОБКА ПРИРОДНОЇ МОВИ, РОЗПІЗНАВАННЯ МОВЛЕННЯ, СИНТЕЗ МОВЛЕННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – архітектура та розробка персонального голосового асистента.

Предмет дослідження – методи і можливості розпізнавання та синтезу мовлення.

Мета роботи – розробка голосового інтерфейсу для інтерактивної та зручної взаємодії україномовних користувачів з власним комп’ютером під керуванням ОС Windows.

Методи дослідження – аналіз існуючих голосових помічників та їх можливостей, аналіз методів та засобів для розпізнавання і синтезу мовлення, програмна реалізація на основі системного аналізу з використанням об’єктно-орієнтованого підходу, опис та методичні вказівки застосування розробленої прикладної програми, обробляння та аналіз отриманого результату.

Під час виконання кваліфікаційної роботи було проведено аналіз щодо предметної області, існуючих голосових помічників та методів і підходів до їх розробки. Було виділено головні недоліки наявних систем та запропоновано своє рішення, яке здатне подолати основні проблеми сучасних голосових асистентів під керуванням операційної системи Windows для україномовних користувачів.

# ABSTRACT

Explanatory note: 62 p., 1 tabl., 17 fig., 1 ann., 11 sources.

ARTIFICIAL INTELLIGENCE, NATURAL LANGUAGE PROCESSING, SPEECH RECOGNITION, SPEECH SYNTHESIS, VOICE ASSISTANT, VOICE INTERFACE.

The object of research is the architecture and development of a personal voice assistant.

The subject of research is the methods and possibilities of speech recognition and synthesis.

The purpose of the work is to develop a voice interface for convenient interaction of Ukrainian-speaking users with their own computer running Windows OS.

Research methods – analysis of existing voice assistants and their capabilities, analysis of methods and tools for speech recognition and synthesis, software implementation based on system analysis using an object-oriented approach, description and methodological guidelines for using the developed application program, processing and analysis of the obtained result.

During the qualification work, an analysis was conducted on the subject area, existing voice assistants, and methods and approaches to their development. The main drawbacks of existing systems were identified, and a proposed solution was put forward that can overcome the main problems of modern voice assistants under the Windows operating system for Ukrainian-speaking users.

**ЗМІСТ**

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API – Application Programming Interface – Прикладний програмний інтерфейс;

ASR – Automatic Speech Recognition – Автоматичне розпізнавання мовлення;

CER – Character error rate – Метрика символьних помилок;

GUI – Graphical user interface – Графічний інтерфейс користувача;

STT – Speech-to-Text – Перетворення мовлення в текст;

TTS – Text-to-Speech – Перетворення тексту на мовлення;

VA – Voice Assistant – Голосовий асистент;

VUS – Voice Usability Scale – Шкала зручності використання голосу;

WER – Word error rate – Метрика помилок у словах.

# ВСТУП

В останні роки віртуальні помічники зі штучним інтелектом стають все більш популярними, займаючи провідне місце нашого повсякденного життя. Завдяки вбудованим алгоритмам машинного навчання та обробки природної мови, ці технології можуть перетворювати рутинні дії на більш зручні та ефективні, забезпечуючи підтримку та спрощуючи виконання завдань.

Голосові помічники інтегровані в різні технологічні пристрої, включаючи смартфони, ноутбуки і настільні комп'ютери, ігрові консолі, телевізори, розумні колонки (аудіосистеми) та автомобілі. Вони здатні використовувати голосову модальність як централізований спосіб зв'язку, що робить графічний інтерфейс користувача непотрібним або менш значущим.

Люди використовують технологію голосових асистентів в різних аспектах свого життя, наприклад для виконання простих завдань, як-от отримання прогнозу погоди, пошук бажаної інформації в інтернеті, керування електронною поштою, та багато інших корисних речей. Це ставить взаємодію людей з обчислювальними системами на новий рівень користування.

Незважаючи на те, що технологія голосових помічників була наявна протягом певного часу, її широке поширення відбулося лише недавно. Першим сучасним та відомим голосовим помічником є Siri від Apple, який був представлений у 2011 році. Невдовзі з'явилися Google Assistant, Alexa від Amazon та Cortana від Microsoft. Зараз на ринку існує велика кількість голосових помічників, і всі вони мають різноманітний функціонал та можливості.

Опитування Capgemini показує, що 76% організацій зафіксували фінансову вигідність від інвестицій у голосові помічники. Компанії вже почали змагання за створення найкращих віртуальних голосових технологій

для їх обслуговування. Наприклад, Amazon Alexa може допомогти людям купувати товари певних брендів завдяки Alexa Skills [1].

В даний час у всьому світі відбувається масове впровадження голосових помічників в різні сфери життя та діяльності людини. Статистичне дослідження зазначає, що тільки в 2020 році було використано та прийнято на обслуговування близько 4,2 мільярди голосових асистентів, з прогнозованим збільшенням до 8,4 мільярдів до 2024 року [2]. Зростання популярності голосових асистентів привело до збільшення обсягу досліджень, які зосереджені на аспектах зручності використання та взаємодії з користувачами.

За допомогою голосових помічників, користувачі можуть виконувати різноманітні завдання без необхідності робити дії власноруч. Це економить час і зусилля та особливо корисно для тих, хто має проблеми з мобільністю або навантажений графік.

Досить широке коло споживачів може використовувати голосові помічники, включно з особами, яким було б важко взаємодіяти через звичайні інтерфейси, такі як клавіатури або сенсорні екрани. Наприклад, голосові асистенти можуть спростити користування пристроями людям з вадами зору.

Важливість зручності використання голосових асистентів було наведено в дослідженні Звакване Д. С., завдяки оцінці VUS (Voice Usability Scale), що базується на якості і актуальності інформації, семантичному інтелекті та задоволеності користувача [3].

Загалом, голосові помічники стають все більше звичними у нашому щоденному житті, і ймовірно, що ця тенденція буде зростати. Для багатьох людей вони стають необхідними, оскільки дозволяють забезпечити новий рівень зручності, автоматизації, доступності та адаптивності.

# 1 АНАЛІЗ ОБРАНОЇ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Місце голосових асистентів в сфері штучного інтелекту

Голосовий асистент (VA, Voice Assistant), який також називають віртуальним голосовим помічником – це комп’ютерна програма, яка здатна розуміти мовлення користувачів і відповідати їм за допомогою синтетичного голосу.

Завдяки новим технологіям, таких як штучний інтелект (ШІ), машинне навчання та обробка природної мови, голосові помічники стають все більш поширеними в людському житті. Передусім вони є інтелектуальними пристроями, здатними самостійно діяти та приймати рішення на основі взаємодії з навколишнім середовищем через голосовий інтерфейс.

Віртуальні помічники діють за допомогою технології синтезу та розпізнавання мовлення, що дозволяє їм пасивно зчитувати всі звукові сигнали, обробляти їх та відповідати на них голосом за необхідності. Користувач називає ключову фразу (наприклад віртуальне ім’я помічника) та додає бажане запитання чи голосову команду. Після цього його запит відправляється на сервер для подальшої обробки даних, що можна поділити на такі умовні етапи:

– сигнал відфільтровується системою шляхом усунення будь-яких звукових шумів та перешкод;

– звукові хвилі очищеного сигналу перетворюються в цифрову форму;

– система проводить аналіз та виділяє ділянки, що містять мовлення, та оцінює їх параметри (наприклад ймовірність кореляції між фразами чи словами);

– визначення найбільш точного шаблону, який відповідає наданому запиту.

Після проходження системою усіх етапів, вона виконує відповідну дію та за потреби надає голосову відповідь. У разі, якщо голосовий віртуальний помічник після обробки даних на сервері не може зрозуміти команду користувача або не може знайти правильну відповідь, то система пропонує користувачеві перефразувати питання або надати додаткову інформацію.

Якщо зануритись в історію, то стане важко повірити, що Siri від Apple не є першим голосовим помічником. Технологія голосового штучного інтелекту існує вже близько 62 років, хоч і не була особливо корисною і широко застосованою в минулі часи. IBM Shoebox, створений у 1961 році, був першим цифровим інструментом для розпізнавання мови. Він міг розпізнавати та реагувати на шість команд, таких як «плюс», «мінус», а також цифри від 0 до 9 [4].

Проте, можливості сучасних інтелектуальних пристроїв є значно глибшими та розвиненішими, порівняно з попередніми системами цієї сфери і напрямів. Функція голосового управління була вперше запроваджена на смартфонах, що дозволила користувачам взаємодіяти з пристроями ефективним та зручним способом. На сьогоднішній день, користувачі цих технологій можуть розмовляти з голосовими асистентами в реальному часі, аналогічно спілкуванню з живими людьми, та отримувати досить чіткі відповіді в межах певного розмовного контексту.

Протягом останніх декількох років, широкої популярності набули розумні колонки з вбудованими голосовими помічниками, адже сервіси та послуги, які вони надають, покращились, що пов’язано зі значними досягненнями в машинному навчанні та обробці природної мови. Таким чином, сучасні голосові асистенти стають більше схожими на людей, що прокладає шляхи в бік нових і футуристичних можливостей технологічного прогресу. Бо вже зараз голосові інтелектуальні системи здатні полегшувати різні завдання повсякденного життя, такі як регулювання яскравості світла в кімнаті, встановлення будильників, надання актуальної інформації про

новини та погоду, допомога в орієнтуванні на дорозі чи будь-якій місцевості, управління електронними листами, та багато інших різноманітних дій.

Голосові помічники широко застосовуються в бізнесі для підвищення точності та продуктивності, допомагаючи усунути чи мінімізувати людські помилки. Наприклад, для полегшення процесів пов’язаних з онлайн-замовленнями. Згідно з декількох досліджень, переважна більшість користувачів веб-сайтів з онлайн-транзакціями віддає перевагу спілкуванню з ботами за допомогою голосу, а не через традиційну систему текстових повідомлень. За результатами опитування встановлено, що впровадження голосових помічників в бізнес-модель компаній дозволяє зекономити до 6,2 мільярдів робочих годин щороку та призводить до величезного збільшення загальної продуктивності бізнесу, що було б неможливим без застосування даних технологій [5].

Дані, які були отримані протягом декількох років, свідчать про те, що споживачі частіше залишають відгук через віртуального помічника, а не вводять текст у відповідні форми власноруч. Застосування голосових асистентів можливе не лише для збору корисних відгуків від клієнтів, але й для планування прибуткових акцій на майбутній фінансовий період спрямованих на цільову аудиторію. Така інформація може бути використана для створення планувальних стратегій з метою максимізації прибутку. Тому саме такий підхід використання голосових помічників надає їм перевагу перед традиційними можливостями.

Ще однією невід’ємною сферою життя людини, де використовуються здібності голосових асистентів є автомобільна промисловість. Потенціал голосових помічників для розмови природною мовою в автомобілях величезний, і завдяки постійним дослідженням і розробкам ми можемо очікувати, що в майбутньому ми побачимо більш вдосконалені та масштабніші можливості цих технологій.

Створення успішного віртуального голосового асистента природною мовою для використання в автомобілях є складним і трудомістким процесом, який потребує кількох ітерацій навчання та тонкого налаштування. Оскільки розробка потребує значної кількості даних, обчислювальних ресурсів і досвіду, лише кілька компаній, таких як Google, Microsoft, NVIDIA, Tesla і Qualcomm, мають ресурси для виконання цієї роботи. Хоча розробка такої системи може зайняти до декількох років, її створення може сприяти збільшенню попиту на транспортні засоби та вплинути на ринок новітніх технологій.

Аналізуючи перспективи та швидкий розвиток сфери штучного інтелекту, можна виділити такі ключові особливості, які притаманні сучасним голосовим асистентам:

– зручність використання. Користувач може давати команди та отримувати відповідь голосом, не вдаючись до таких інтерфейсів взаємодії, як клавіатура чи сенсорні екрани;

– автоматизація. Голосові асистенти здатні автоматизувати багато процесів, таких як пошук інформації, керування побутовими приладами, встановлення нагадувань і т. д.;

– швидкодія та ефективність. Персональні помічники допомагають виконувати дії з мінімальною затратою часу та сил, за рахунок паралельного і асинхронного виконання декількох поставлених завдань;

– навчання та персоналізація. Розумні системи можуть навчатися розпізнавати голос користувача, враховувати його індивідуальні потреби та вподобання;

– доступність. Голосові помічники дозволяють людям з різними фізичними обмеженнями і вадами використовувати сучасні технології та долучатись до цифрового світу.

## 1.2 Огляд існуючих віртуальних голосових помічників

На сьогоднішній день існує досить велика кількість голосових асистентів, які відрізняються між собою за такими властивостями: сфера і область застосування, наявність певних можливостей та функціоналу, підтримка різних локалізацій, вартість використання та доступність. Було розглянуто та проаналізовано можливості найпопулярніших віртуальних помічників, як-от Cortana, Mycroft, Alexa, Braina, що здатні функціонувати на комп'ютерах під керуванням актуальних версій операційної системи Windows.

Більшість людей, які використовували Windows 10 чи 11 з високою долею ймовірності хоча б один раз за своє життя чули про віртуальний помічник Cortana, який вбудовано в ці операційні системи. Це корисний інструмент на основі штучного інтелекту та пошукової системи Bing, який можна використовувати для диктування електронних листів, налаштування нагадувань чи заміток, пошуку різноманітної інформації в Інтернеті, управління файловою системою та багатьох інших задач.

Голосовий асистент Cortana було вперше представлено в 2014 році та розроблявся виключно для Windows Phone 8.1. Однак пізніше його було інтегровано з іншими пристроями та операційними системами, такими як Windows 10 і 11, Android, iOS та системним програмним забезпеченням Xbox One. Також цифровий помічник здатний працювати з пакетом продуктів Microsoft 365 та підтримує браузер Microsoft Edge.

Cortana ґрунтується на технологіях машинного навчання та штучного інтелекту, які дозволяють йому з часом розуміти вподобання, симпатії, антипатії та звички користувача. Крім того, асистент веде інтелектуальні діалоги з користувачами, а не просто відтворює попередньо збережений набір запитань та відповідей.

Звичайно, щоб Cortana працювала належним чином, їй потрібен доступ до всієї інформації на комп'ютері користувача. Цифровий помічник

може використовувати всі ці дані для створення своєчасних нагадувань, роблячи відповідні припущення та надаючи всю необхідну інформацію. Деякі люди можуть вважати, що це впливає на їх конфіденційність, тому не наважуються на використання можливостей та функцій помічника Cortana.

Mycroft – це голосовий помічник з відкритим кодом, який можна встановити на різних пристроях, таких як смартфони та комп'ютери. Mycroft розроблено таким чином, щоб бути зосередженим на конфіденційності. Тому він є переконливою альтернативою фірмовим помічникам зі штучним інтелектом, таким як Cortana, оскільки не збирає та не зберігає особисті дані споживачів цього продукту.

Однією з унікальних особливостей Mycroft є його гнучкість. Його доступність з відкритим кодом дозволяє розробникам налаштовувати помічника відповідно до своїх потреб, а користувачам обирати, які функції ввімкнути або вимкнути залежно від своїх уподобань.

Голосовий асистент від компанії Mycroft AI не має офіційної підтримки операційної системи Windows, але дозволяє скористатись його можливостями за рахунок використання віртуальної машини VirtualBox, що може відлякувати необізнаних користувачів комп'ютерів. Mycroft має меншу базу споживачів порівняно з Cortana та не має такої ж кількості функцій та інтеграцію з іншим програмним забезпеченням і службами.

Загалом, Mycroft є перспективним помічником з відкритим кодом, який пропонує альтернативу пропрієтарним голосовим асистентам сучасного ринку сфери штучного інтелекту. Його гнучкість і підхід, орієнтованість на конфіденційність, роблять його привабливим варіантом для користувачів, які цінують налаштування та безпеку даних.

Alexa – віртуальний голосовий помічник, розроблений компанією Amazon та базується на технологіях викупленого стартапу польської компанії Ivona в 2013 році. Він здатний виконувати різноманітні завдання, зокрема відтворювати музику, відповідати на запитання, встановлювати

будильники, керувати пристроями розумного дому та навіть здійснювати покупки через Amazon.

Основною особливістю помічника Alexa є здатність інтегруватися з великою кількістю розумних домашніх пристроїв, включаючи Amazon Echo, Dot, Show та Fire TV. Це означає, що користувачі можуть керувати будь-чим: від світильників і термостатів до дверних замків і систем безпеки, використовуючи для цього лише голосові команди. Крім того, Alexa має зростаючу бібліотеку з понад 100 000 навичок та умінь, які активно створюються і поліпшуються сторонніми розробниками та дозволяють використовувати голосового помічника для виконання таких різноманітних завдань, як замовлення їжі чи наприклад взаємодії з комп'ютерними іграми.

Alexa Guard – це корисна функція безпеки, яка надійно відслідковує діяльність всередині та за межами будинку користувачів, коли вони мають бути відсутні впродовж деякого часу. Якщо розумний динамік Echo вловить незвичні звуки, як-от розбиття скла чи інший дивний шум, Alexa швидко попередить власників сповіщенням для перевірки підозрілої ситуації. З Alexa Guard Plus (платна версія базової функції) користувачі матимуть такий самий функціонал моніторингу будинку, але з доступом до екстреної гарячої лінії Amazon та додаткових інструментів безпеки.

У порівнянні з Cortana, голосовий асистент від Amazon має досить значний набір можливостей та більшу бібліотеку сторонніх функцій. Проте Cortana має перевагу в плані інтеграції з пакетом продуктів компанії Microsoft, таким як Office та ін.

Голосовий асистент Alexa не потребує складного налаштування перед початком ознайомлення з його особливостями. Для цього лише потрібно завантажити офіційну програму Alexa з Microsoft Store та почати користування, слідуючи додатковим підказкам, які надає асистент. На рисунку 1.1 наведено вигляд прикладної програми голосового помічника Alexa та деяких функціональних можливостей.

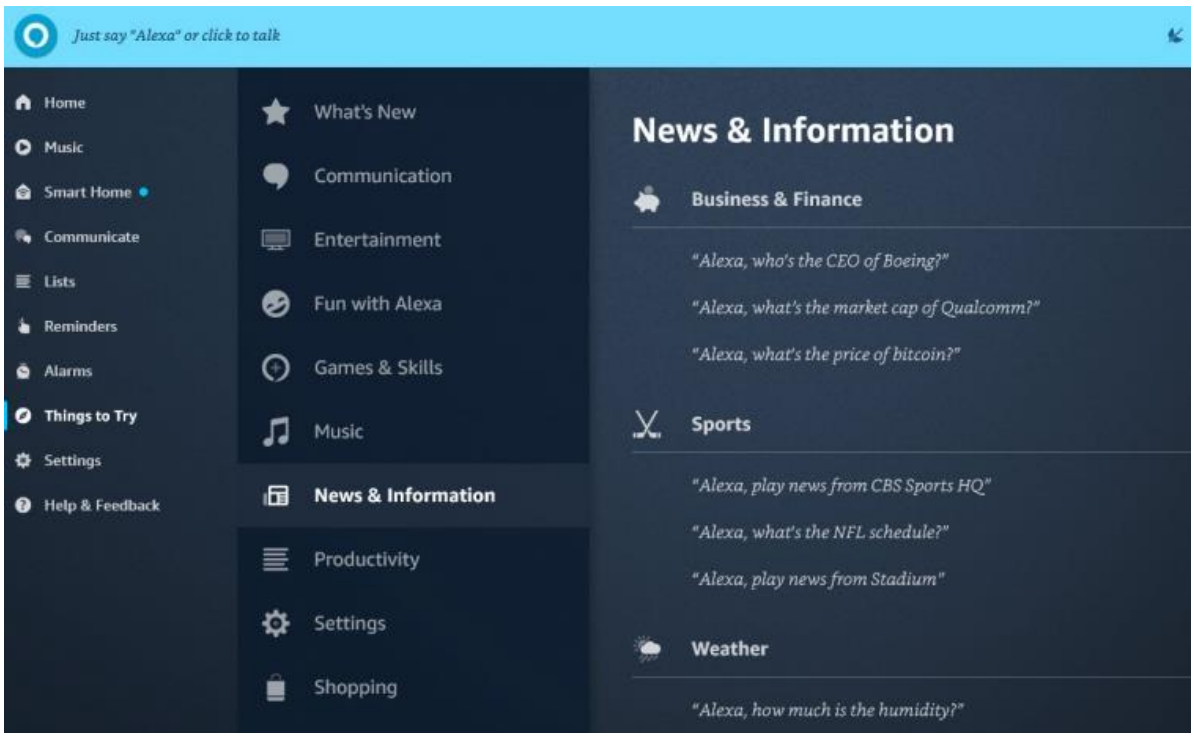


Рисунок 1.1 – Функціональні можливості Alexa

Amazon намагається постійно поліпшувати і розширювати можливості та функції Alexa, створюючи цінні технології, які буде легко та зручно застосовувати. Наразі компанія активно розробляє функції виявлення емоцій, щоб Alexa могла зрозуміти настрій користувачів та розпізнати, чи розчаровуються вони в її роботі.

Braina – це інноваційний голосовий асистент, який забезпечує ефективний і персоналізований досвід для своїх користувачів. Він є впевненою альтернативою Cortana для Windows 10 і має чисельну функціональність, що робить його кращим вибором для деяких користувачів.

Однією з ключових особливостей помічника Braina є точність розпізнавання голосу, яка є однією з найкращих на ринку сьогодення. Це надає можливість користувачам взаємодіяти з комп’ютером за допомогою розмовної мови, а голосовому асистенту розуміти складні команди, що робить його зручним для професійного використання.

Список команд Braina різноманітний, що дозволяє користувачам відкривати програми, виконувати веб-пошук, встановлювати нагадування чи будильники, керувати вікнами та інтерфейсом, допомагати з вирішенням математичних завдань, зачитувати інформацію із веб-сторінок, перекладати текст між різними мовами та з легкістю управляти функціями комп'ютера.

На відміну від Cortana, Braina не обмежується лише Windows 10 і може використовуватися на інших платформах, таких як Android та iOS. Ця функція робить Braina чудовим вибором для людей, які використовують кілька пристроїв і потребують плавного переходу між ними.

Налаштування функцій Braina – це ще один аспект, який відрізняє його від аналогів. Користувачі можуть налаштовувати мову, акцент та поведінку асистента, що дозволяє створити унікальний досвід використання програми. Цей рівень налаштування дозволяє користувачам адаптувати помічника до своїх індивідуальних потреб і вподобань.

Незважаючи на вказані переваги, Braina все ж має деякі недоліки. Одним із найбільш суттєвих недоліків, порівняно з Cortana – це відсутність інтеграції з продуктами Microsoft Office, що може стати перешкодою для деяких споживачів цього продукту. Крім того, безкоштовний функціонал програми доволі обмежений, і вимагає додаткової плати за повну версію, щоб отримати доступ до більших можливостей використання.

Проблемою, якою можна охарактеризувати більшість сучасних голосових асистентів, є конфіденційність та безпека даних, адже компанії можуть збирати та обробляти всю необхідну для них вхідну інформацію людей, та вимагає постійного підключення до мережі Інтернет. Плюс до цього, компанії прагнуть збирати статистику запитів користувачів. Вони зберігають показники, щоб визначити, чим люди цікавляться, що найчастіше шукають, яким послугам надають перевагу, яку музику слухають і т. д. Інформація є анонімною лише в мінімальній мірі, яка вимагається законом. Ці записи можуть містити дані, які вказують на місцезнаходження чи контактну інформацію користувачів.

Навіть якщо записи деяких людей не були серед цієї частини проаналізованих даних, вони все одно знаходяться на серверах. Корпорація Майкрософт дозволяє собі отримувати доступ до розшифровок голосів без згоди із споживачами. Apple зберігає стенограми того, що диктують користувачі. Amazon зберігає велику кількість аудіоінформації: один репортер Reuters з'ясував, що протягом чотирьох років Alexa зробила понад 90 000 записів його даних, включаючи членів його родини [6].

Компанії не здатні легко відмовитись від збору інформації про користувачів, адже чим більшою інформацією та пов'язаними даними вони володіють, тим більший прибуток вони отримують, продаючи цю інформацію рекламодавцям.

Головну проблему, яку також слід виділити в більшості голосових асистентів, включаючи ті, що мають можливість функціонувати під керуванням операційної системи Windows, є відсутність нативної української локалізації та офіційної підтримки користувачів з України, що може бути обумовлено недостатньою ринковою потребою та наявністю необхідних ресурсів для цього. Однак з часом, попит на українську локалізацію зросте та зможе спонукати компанії додати її у майбутніх версіях своїх продуктів.

# 2 АНАЛІЗ МЕТОДІВ ДЛЯ СТВОРЕННЯ ПЕРСОНАЛЬНОГО ГОЛОСОВОГО АСИСТЕНТА

## 2.1 Основні підходи до розробки голосового асистента

Стрімкий розвиток технологій, безперечно, відкриває нові можливості для користувачів, але також може бути викликом для розробників. У найближчому майбутньому наявність голосового інтерфейсу в програмі може вважатися обов'язковою умовою. Тому слід розібратись, які є основні принципи та підходи до розробки віртуального голосового помічника.

Можна виділити три основні методи, які використовуються для розробки персональних голосових помічників:

– швидкий метод: передбачає інтеграцію готових голосових асистентів (таких як Google Now, Siri) в бажану програму;

– збалансований метод: для цього потрібні базові знання технології машинного навчання та створення розумного помічника з використанням готових служб з відкритим кодом та API. Серед таких інструментів, можна виділити Melissa, Api.ai, Dialogflow, Jasper і т. п.;

– просунутий метод: розробка усіх ключових компонентів голосового асистента з нуля, мінімізуючи використання сторонніх рішень, що вимагає значної кількості ресурсів та знань в цій області дослідження.

Перший метод розробки має ряд певних недоліків, а саме: обмежений доступ до функціональних можливостей та строге дотримання умов під час використання доступних SDK, постійне підключення програми до мережі Інтернет, порушення проблеми конфіденційності та безпеки даних, слабка персоналізація, відсутність глибокого налаштування і т. д.

Багато людей віддають перевагу використанню знайомих та надійних технологій для взаємодії. Однак, мобільні помічники мають свої обмеження щодо інтеграції зі сторонніми додатками та функціональністю, яка може

сильно відрізнятись на деяких платформах. Тому ці фактори перешкоджають гнучкості розвитку.

Сервіси, які були згадані в другому методі, дозволяють розробникам отримати готові рішення основних компонентів для функціонування голосового асистента за допомогою відкритого коду та API. Проте більшість сервісів мають обмежений безкоштовний функціонал, не повну підтримку української локалізації та наслідують деякі проблеми першого методу.

Альтернативні рішення значно спрощують процес впровадження. Дотримуючись їхніх інструкцій, можна без жодних зусиль ввімкнути розпізнавання чи синтезу голосу у власній програмі. Однак це не буде мати свободи внесення значних змін та додаткових функцій.

Головна перевага самостійної розробки полягає в тому, що ми вільні реалізовувати все, що забажаємо, але витрачаючи на це досить значну кількість ресурсів. Тому оптимальним рішенням для створення персонального голосового асистента, є поєднання збалансованого та просунутого методу, використовуючи добре підібрані технології та рішення з відкритих джерел.

## 2.2 Архітектурні властивості голосового асистента

Голосовий асистент може містити доволі широкий спектр технологій штучного інтелекту для належного функціонування, проте слід виділити дві, які є ключовими компонентами його архітектури: розпізнавання мовлення та синтез голосу.

Перетворення аудіофайлів або голосового введення з мікрофона на текст відоме як STT (speech-to-text). STT має «сприймати» вхідне аудіо (голос людини) та «розпізнавати» вимовлені слова для отримання повного текстового запиту. Протилежним до перетворення мовлення в текст є TTS (text-to-speech), процес, який моделює природну мову та перетворює текст

у аудіо для подальшого відтворення. На рисунку 2.1 наведено базову архітектуру віртуального голосового помічника [7].

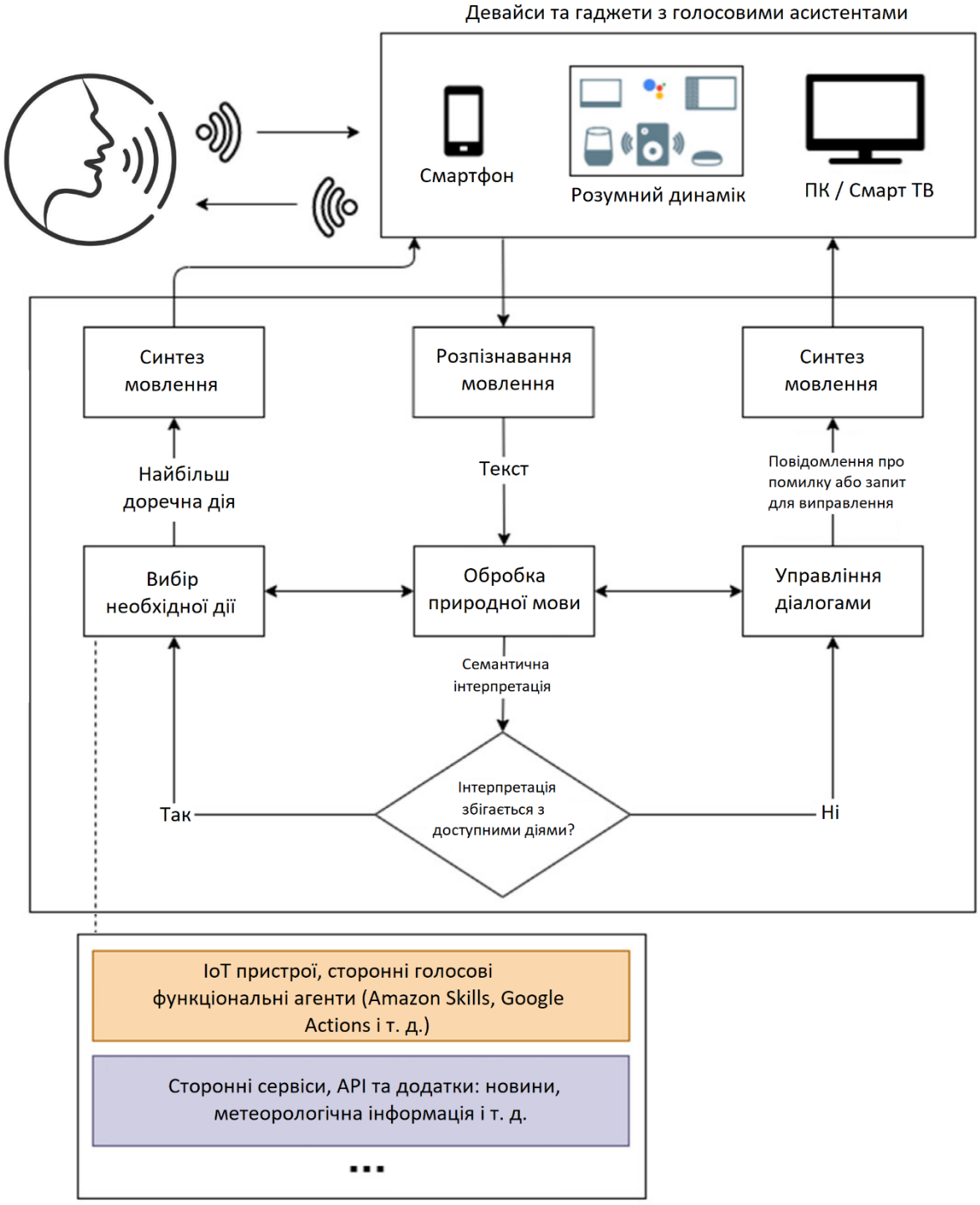


Рисунок 2.1 – Базова архітектура віртуального голосового асистента

### 2.2.1 Розпізнавання мовлення

Розпізнавання мовлення, яке також називають автоматичним розпізнаванням мовлення (ASR), комп’ютерним розпізнаванням мовлення або перетворенням мови в текст (STT), є формою штучного інтелекту та стосується здатності комп’ютера або машини інтерпретувати вимовлені слова та переводити їх у текст.

Історія розвитку розпізнавання мовлення охоплює декілька десятиліть і починається з розробки перших систем розпізнавання мовлення на початку 1950-х років. Протягом наступних десятиліть розробники працювали над різними методами та технологіями, щоб поліпшити точність розпізнавання та забезпечити більш широку підтримку різних мов та акцентів.

У 1990-х роках вчені стали використовувати нейронні мережі для розпізнавання мовлення. Ці системи мали більшу точність та могли розпізнавати більш широкий діапазон мов та акцентів. Проте, вони потребували значної обчислювальної потужності та були дуже дорогими.

У 2000-х роках з’явилися системи розпізнавання мовлення на основі глибокого навчання, які використовуються понині. Ці системи мають велику точність та можуть розпізнавати різні мови та акценти з високою швидкістю. Крім того, вони можуть працювати на звичайних смартфонах та інших пристроях.

Розпізнавання мовлення та голосу досить тісно пов’язані, та часто використовуються разом у спільних системах. Однак, ці технології відрізняються та використовуються для різних задач.

Розпізнавання мовлення означає процес розпізнавання, розуміння та транскрибування мовлення комп’ютером у письмовий текст, який можна прочитати. Технології розпізнавання мовлення аналізують акустичні характеристики звукових і голосових сигналів, такі як висота, темп, різні акценти та інші мовні змінні, щоб ідентифікувати та перетворювати послідовності слів у повноцінний текст.

Натомість, розпізнавання голосу перетворює голос у цифрові дані на основі унікальних голосових характеристик користувача. Ця технологія являє собою біометричну систему, яка використовується для перевірки особи людини шляхом аналізу унікальних особливостей її голосу, таких як висота, тон і ритм. Розпізнавання голосу часто використовується для безпеки та особистої автентифікації, наприклад для розблокування мобільного пристрою або доступу до персональних даних чи документів.

Людське мовлення є одним із найскладніших предметів для вивчення та включення в сучасні цифрові технології. Існує багато мовних змінних, які слід враховувати, тому створення технології для правильного розуміння, аналізу та отримання відповідних результатів завжди було величезним викликом.

Більшість технологій розпізнавання мовлення сьогодення, оцінюються за показником точності або коефіцієнтом помилок у словах (WER). WER – кількість помилок, поділена на загальну кількість слів. Щоб обчислити WER, потрібно підсумувати заміни, вставки та видалення, які відбуваються в розпізнаній послідовності слів, а потім розділити це число на загальну кількість слів, які були сказані напочатку.

Існує також багато факторів, які можуть впливати на точність, наприклад акцент, висота звуку, вимова, гучність і фоновий шум. Завдяки цьому існує багато алгоритмів, які були винайдені для досягнення кращих результатів у розпізнаванні мовлення, наприклад ПММ (прихована марковська модель), на якому базується більшість сучасних систем розпізнавання мовлення.

Також, обов’язковою умовою для використання розпізнавання мовлення є врахування основних властивостей вхідного аудіо, а саме: формат, кількість каналів (стерео чи моно), частоту дискредитації, бітрейт, тривалість. Найважливішими з них є формат, кількість каналів та частота дискретизації аудіофайлу.

Деякі моделі є стійкими до змін цих параметрів, однак є й такі, які можуть сприймати лише обмежену кількість, тому потрібно враховувати, що більшість моделей були навчені на наборах даних із вибірками 16Гц та одним каналом (моноаудіо). Щоб мінімізувати цей вид неузгодженості, можна використовувати різні методи для роботи зі звуком у кожній мові програмування. Наприклад, у Python різні операції, такі як читання, перетворення і запис аудіо, можна виконувати за допомогою таких бібліотек, як Librosa, PyDub, scipy.io.wavfile, playsound та інші.

Python налічує досить значну кількість фреймворків та бібліотек які дозволяють використовувати технологію розпізнавання мовлення для розробки власного голосового асистента, а саме: Microsoft Azure Speech Services, Google Cloud Speech-to-Text API, pocketsphinx, Mozilla DeepSpeech, Nvidia Nemo, Kaldi, SpeechRecognition та інші.

### 2.2.2 Синтез мовлення

Перетворення тексту в мовлення (Text-to-Speech, TTS) – це технологія синтезу мовлення, яка здатна перетворити друкарський текст на голосовий сигнал. У 1990-х роках синтез мовлення став доступним для широкого кола користувачів завдяки появі програмного забезпечення для персональних комп’ютерів. У 2000-х роках віртуальні помічники стали все більш популярними, що призвело до появи нових технологій синтезу мовлення.

На сьогоднішній день синтез мовлення став все більш точним та надійним завдяки розвитку штучного інтелекту та обробки природної мови. Його застосування охоплює багато сфер, включаючи технології підтримки відновлення мовлення для людей з різними видами порушень мовлення, відеоігри, віртуальних асистентів, рекламні системи, автоматизовані телефонні системи та інші.

Одним з ключових викликів для розвитку синтезу мовлення є покращення природності та емоційності голосу, що дає змогу зробити

відтворення мовлення більш приємним для слухачів. Це досягається застосуванням глибокого навчання та нейромереж. Глибокі нейронні мережі зробили революцію в технології синтезу мовлення, дозволивши штучному голосу звучати так само природно й реалістично, як людське мовлення.

Інший виклик пов’язаний з мовленням, який ще потребує більшого розвитку – це можливість синтезу мовлення на інших мовах, окрім англійської, та на мовах з різними діалектами та акцентами. Це потребує створення більш складних та точних моделей, які здатні враховувати особливості кожної мови та акценту.

Всього можна виділити три основні типи TTS:

– синтез конкантинацією;

– статистичний параметричний синтез;

– нейронний синтез.

Синтез конкантинацією засновується на об’єднані готових фрагментів мовлення, які зберігаються в базі даних. В базі зазвичай містяться окремі фрагменти голосів, починаючи від повних речень і закінчуючи окремими складами. Необхідність великої бази даних з різними комбінаціями є вагомим недоліком цього методу.

Статистичний параметричний синтез мовлення (Statistical Parametric Speech Synthesis, SPSS) зазвичай складається з трьох компонентів: модуля аналізу тексту, акустичної моделі (модуль прогнозування параметрів) та аналізу/синтезу вокодера. Модуль аналізу тексту спочатку обробляє текст, а потім витягує лінгвістичні характеристики, такі як фонеми, POS-теги (наприклад іменники, прикметники, і т. д.) та тривалість. Акустична модель обробляє лінгвістичні дані для створення акустичних характеристик, які потім обробляються вокодером для генерації хвилі.

Нещодавні вдосконалення підходів на основі нейронної мережі (NN) призвели до їх використання у технологіях TTS. Зазвичай ці моделі використовують будь-який із двох підходів: замінюють один або багато

компонентів SPSS відповідними NN моделями або використовують модель на основі наскрізної нейронної мережі для заміни всіх компонентів SPSS.

Під час навчання глибока нейронна мережа отримує записаний голос та відповідний текстовий варіант, щоб потім виявити відношення між одним бітом тексту та пов’язаними акустичними характеристиками. Тому нейронний синтез можна сміло вважати рушійною силою прогресу в технологіях TTS, що є ключом до більш точних та реалістичних результатів.

Python є однією з найпопулярніших мов програмування в світі та сфері штучного інтелекту і має велику кількість бібліотек та фреймворків для різних задач. Для синтезу мовлення також існує чимало інструментів, які дозволяють розробникам створювати високоякісні голосові додатки, серед яких можна виділити наступні: Kaldi, Festival, ukrainian-tts, gTTS, edge-TTS, pyttsx3, RHVoice, ESPNet, espeak-ng.

# 3 ПОСТАНОВКА ТА ОПИС СТРУКТУРНИХ І ФУНКЦІОНАЛЬНИХ ОСОБЛИВОСТЕЙ ЗАДАЧІ

Проаналізувавши існуючі голосові асистенти, які здатні функціонувати під керуванням актуальних версій операційної системи Windows, можна виділити такі їх основні недоліки:

– відсутність української локалізації та підтримки користувачів з України;

– порушення проблеми конфіденційності та безпеки даних;

– наявність постійного підключення до мережі Інтернет.

Тому головним завданням магістерського проєкту є: розробка програмного модулю голосового асистента для синтезу та розпізнавання українського мовлення.

Метою рішення даного завдання є можливість інтерактивної та зручної взаємодії україномовних користувачів з власним комп'ютером під керуванням ОС Windows, використовуючи голосовий інтерфейс.

Для досягнення поставленої мети слід розглянути наступний набір питань:

– проаналізувати доступні моделі синтезу та розпізнавання українського мовлення;

– обрати найдоречніші методи та технології для розробки;

– програмно реалізувати основну архітектуру голосового асистента;

– зробити тестування та перевірку функціональних можливостей створеного програмного продукту.

Технічні вимоги, яким має задовольняти розроблений голосовий асистент:

– підтримка функціонування у операційних системах Windows 10 та 11;

– забезпечувати повну конфіденційність та безпеку персональних даних користувачів;

– дотримуватись законодавства щодо збору та обробки вхідної інформації;

– здатність ефективного використання ресурсів пристрою;

– виконувати розпізнавання та синтез в локальній системі, за рахунок обчислювальних ресурсів комп’ютера.

Функціональні особливості і вимоги, які має забезпечувати розроблений голосовий асистент:

– наявність нативної української локалізації;

– мати зрозумілу та просту взаємодію з користувачем;

– приймати голосову команду користувача через мікрофон;

– розпізнавати мовлення в реальному часі;

– виконувати відповідну дію на запит користувача;

– надавати голосову відповідь за необхідності;

– здатність функціонувати без доступу до мережі Інтернет;

– мати здатність до масштабування та легкого впровадження нових функціональних можливостей.

Для використання розробленого голосового помічника, комп’ютер має відповідати наступним рекомендованим технічним вимогам:

– операційна система: Windows 10 або вище, 64-бітна версія;

– процесор: Intel Core i3-2100 та його аналоги за продуктивністю;

– оперативна пам’ять: не менше 4 ГБ;

– має мати вбудовану звукову карту та мікрофон, або підключені зовнішні аналоги;

– інтернет-підключення: необхідне тільки для функціональних можливостей (наприклад для отримання погодних даних через API).

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПЕРСОНАЛЬНОГО ГОЛОСОВОГО АСИСТЕНТА

## 4.1 Вибір необхідних технологій для створення віртуального голосового помічника

Ключовою складовою під час розробки будь-якої програми є вибір мови програмування, тому для написання коду власного голосового асистента було обрано мову Python.

Python – це інтерпретована мова програмування, яка використовується для розробки широкого спектру програмного забезпечення, від веб-додатків до штучного інтелекту. Основна філософія мови полягає в тому, що код повинен бути легким для читання та розуміння, навіть якщо потрібно написати значну кількість рядків коду. До ключових особливостей мови програмування Python, можна віднести наступне:

– крос-платформенність: Python працює на багатьох операційних системах, включаючи Windows, macOS та Linux;

– має досить велику та активну спільноту розробників, які створюють та підтримують багато корисних бібліотек та модулів;

– має різноманітний спектр застосування та використовується для розробки веб-додатків, ігор, наукових досліджень, штучного інтелекту, машинного навчання, обробки даних та багатьох інших проектів;

– має багато інтегрованих функцій для обробки рядків, математичних операцій, роботи з файлами, мережевими протоколами і т. д.;

– налічує багато корисних бібліотек, таких як NumPy для наукових досліджень та обробки даних, TensorFlow для штучного інтелекту та машинного навчання, Django для розробки веб-додатків та багато ін.

Python є інтерпретованою мовою, а отже програми можуть бути запущені без попередньої компіляції. Це дозволяє зменшити час розробки та спростити налагодження програм. Іншою важливою особливістю є те, що

Python є мовою з відкритим вихідним кодом. Це означає, що будь-хто може змінювати та розповсюджувати код на основі цієї мови, що сприяє швидкому розвитку та поширенню інновацій.

Python відіграє значну роль в сфері штучного інтелекту (AI), машинного навчання (Machine Learning) та глибинного навчання (Deep Learning). Це зумовлено тим, що він є потужною та гнучкою мовою програмування, яка має багато інтегрованих і сторонніх бібліотек та інструментів для роботи з даними та машинним навчанням, що дозволяє виконувати дослідження в цій сфері легко та ефективно.

В якості середовища для написання програмного коду голосового асистента, було обрано Microsoft Visual Code. VS Code є одним з найпопулярніших та найбільш розповсюджених редакторів для розробки програмного забезпечення, в тому числі й для розробки програм мовою Python.

По-перше, VS Code є дуже легким та швидким редактором, має дуже зручний інтерфейс та дозволяє розробникам досягти високого рівня продуктивності і зосередитись на кодуванні без зайвих відволікаючих факторів.

По-друге, VS Code має вбудовану підтримку для багатьох мов програмування та платформ, включаючи JavaScript, Python, PHP, Java, C++, і багато інших. Це робить його дуже універсальним та зручним для розробки будь-якого виду програмного забезпечення.

По-третє, VS Code має велику кількість сторонніх безкоштовних плагінів, що дозволяє розширити його функціональність та відповідати на всі потреби будь-якого розробника. Завдяки цьому редактор з легкістю можна налаштувати відповідно до своїх вподобань.

Однією з найпопулярніших бібліотек для машинного навчання є PyTorch, тому її використання є невід’ємним фактором для належного функціонування розроблюваного голосового помічника. PyTorch – це відкрите програмне забезпечення для машинного навчання, яке було

розроблено на мові програмування Python командою дослідників з Facebook AI Research (FAIR) в 2016 році. PyTorch має декілька особливостей, які роблять його популярним серед дослідників та практиків машинного навчання:

– має простий та інтуїтивно зрозумілий інтерфейс для використання. Це зроблено для того, щоб дослідники та практики машинного навчання могли швидко розробляти та тестувати свої ідеї;

– надає можливість автоматичного обчислення похідних функцій, що дозволяє реалізувати алгоритми зворотнього поширення помилки без необхідності вручну обчислювати похідні;

– є надійним та швидким фреймворком машинного навчання. Він може бути використаний для вирішення різноманітних задач, від класифікації зображень до обробки природних мов;

– має багато бібліотек, які можуть бути використані для розширення його функціональності. Ці бібліотеки можуть допомогти з навчанням глибоких нейронних мереж, розпізнання об’єктів, обробки природних мов, рекомендаційних систем та багатьох інших завдань машинного навчання;

– фреймворк може працювати з різними пристроями, включаючи CPU та GPU. Це дозволяє розпаралелювати обчислення, що забезпечує більш швидку обробку даних;

– має вбудовані техніки регуляризації та оптимізації, які допомагають знизити перенавчання та забезпечують більш точні результати;

– співпрацює з іншими бібліотеками машинного навчання, такими як TensorFlow, Keras та MXNet. Це дає можливість розширити функціональність PyTorch та використовувати його в різних проектах машинного навчання та багато ін.

Проаналізувавши все це, можна впевнено сказати, що PyTorch є потужним та гнучким фреймворком машинного навчання, який дозволяє розробляти та навчати глибокі нейронні мережі та виконувати складні обчислення з високою швидкістю та точністю.

Хоча користувач має взаємодіяти з віртуальним асистентом завдяки голосовим командам, наявність мінімального інтуїтивно-зрозумілого графічного інтерфейсу буде забезпечувати зручність під час користування розробленою прикладною програмою. Тому для розробки десктопного додатку і всіх його графічних складових було обрано бібліотеку PyQt5.

PyQt5 – це бібліотека Python, яка надає інструментарій для створення графічних інтерфейсів користувача (GUI) на базі фреймворку Qt. Qt – це крос-платформений фреймворк для створення програмного забезпечення, який підтримує розробку програм для різних операційних систем, таких як Windows, MacOS та Linux. PyQt5 містить набір класів, які дозволяють створювати вікна, кнопки, меню та інші елементи інтерфейсу, надає можливість обробки подій, які виникають при взаємодії користувача з програмою, та реалізовувати багато інших функцій, необхідних для розробки сучасних додатків з графічним інтерфейсом.

PyQt5 має підтримку багатопоточності, що дозволяє розробникам створювати додатки, які можуть працювати з декількома потоками одночасно. Однією з особливостей багатопоточності в PyQt5 є те, що вона дозволяє взаємодіяти з графічним інтерфейсом з різних потоків, що робить її особливо корисною для створення складних та масштабованих додатків. Потоки можуть виконувати різні завдання, такі як завантаження даних з Інтернету або обчислення складних операцій, не блокуючи основний потік, який відповідає за роботу з графічним інтерфейсом.

## 4.2 Розробка ключових модулів та компонентів голосового асистента

Для написання прикладної програми голосового асистента використовувалася мова програмування Python версії 3.8.16 та було встановлено всі необхідні бібліотеки і залежності в окремому віртуальному середовищі під керуванням менеджеру пакетів Conda.

Завдяки віртуальним середовищам, можна ізолювати залежності проекту один від одного, та запобігти конфліктам і помилкам, тому що різні проекти можуть вимагати різні версії пакетів та бібліотек. Крім того, віртуальні середовища, які були створені за допомогою Conda, можуть бути легко відтворені на інших системах, що дозволяє легко переносити свій код між різними комп'ютерами та операційними системами.

Проект розробленої прикладної програми налічує чотири Python файли, шість зображень, дві окремі моделі та один файл json. На рисунку 4.1 наведено файлову архітектуру проекту розробленого голосового асистента.

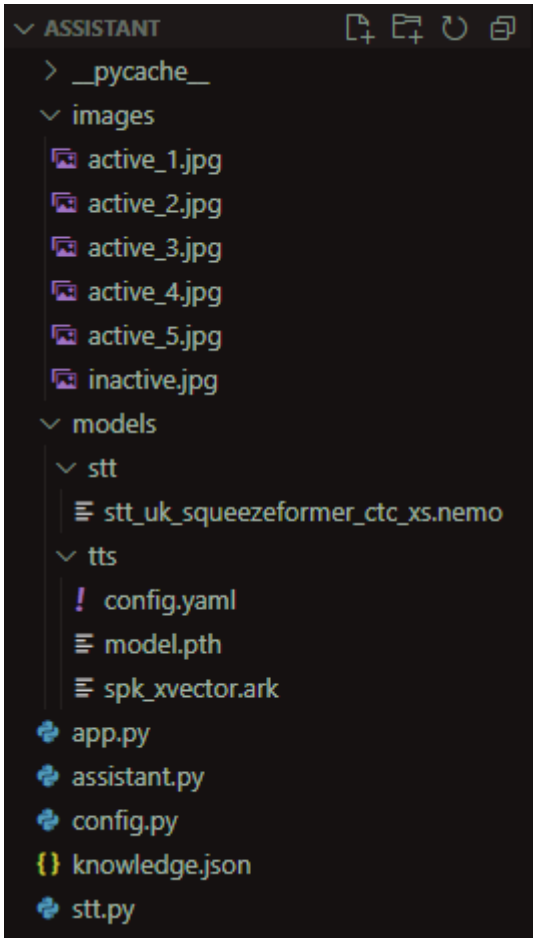


Рисунок 4.1 – Файлова архітектура проекту розробленого голосового асистента

Хоч Python і дозволяє писати код для програм в стилі функціонального програмування, все ж для розробки власної системи голосового помічника було надано перевагу в об’єктно-орієнтованому підході (ООП). Основною ідеєю ООП є те, що програмне забезпечення повинно бути створене у вигляді об’єктів, кожен з яких має свої властивості та методи. Об’єкти можуть взаємодіяти між собою, передавати дані та виконувати відповідні дії.

ООП дозволяє створювати великі та складні програми та розбивати їх на менші взаємопов’язані частини, якими можна легше керувати та проводити тестування. Окрім цього, ООП забезпечує кращу повторну використовуваність коду, що скорочує час розробки та сприяє зменшенню кількості помилок у програмі.

Іншим важливим поняттям ООП є наслідування, що дозволяє створювати нові класи, які успадковують властивості та методи від батьківських та надає можливість створювати більш спеціалізовані класи.

Програма розробленого голосового асистента налічує такі основні класи: MainWindow, Assistant, STT та TTS. Вхідною точкою прикладної програми є клас MainWindow, який відповідає за графічне представлення та оперує класом Assistant, що в свою чергу містить модулі розпізнавання (STT) та синтезу мовлення (TTS).

На рисунку 4.2 наведено діаграму класів, на якій зображено три власні класи та один сторонній пов’язаний клас TTS, бібліотеки ukrainian-tts – для відображення повноцінної архітектури створеного віртуального помічника.

Діаграма класів – це один із основних типів діаграм в о’єктно-орієнтованому програмуванні (ООП), яка використовується для відображення класів, інтерфейсів та їх відносин у системі. Вона визначає структуру системи та допомагає візуалізувати, як класи взаємодіють один з одним.

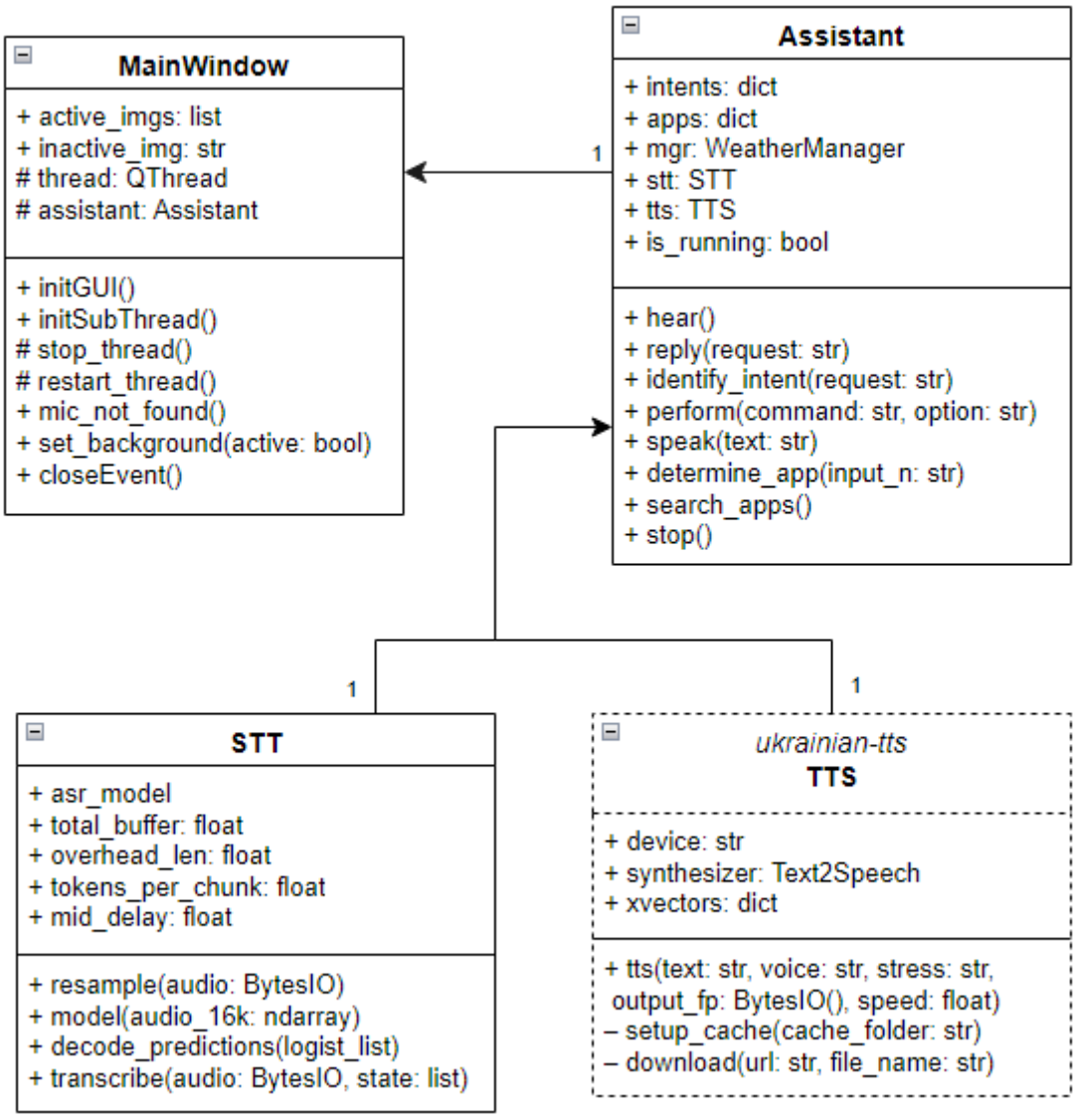


Рисунок 4.2 – Діаграма класів розробленої прикладної програми віртуального голосового асистента

Клас MainWindow описує поведінку графічного інтерфейсу головного вікна програми та здійснює керування допоміжним потоком в якому виконується увесь функціональний процес віртуального асистента. Завдяки цьому, виконання додаткового потоку не буде блокувати та перешкоджати правильному відображенні основного вікна і його графічних елементів.

MainWindow успадковується від класу QMainWindow бібліотеки PyQt5 та має такі атрибути: публічний active_imgs з типом list, публічний inactive_img з типом str, захищений thread з типом QThread, захищений

assistant з типом Assistant. Клас MainWindow містить наступний перелік власних методів:

– initGUI(): конструктор для ініціалізації графічних елементів і властивостей;

– initSubThread(): конструктор для ініціалізації та початкового запуску додаткового потоку в якому буде виконуватись робота процесів голосового асистента;

– _stop_thread(): захищений метод для надійного закриття додаткового потоку;

– _restart_thread(): захищений метод для перезапуску додаткового потоку;

– mic_not_found(): публічний метод для обробки відсутності доступного мікрофону;

– set_background(active: bool): публічний метод для зміни візуального стану активності голосового асистента;

– closeEvent(): метод для коректного закриття програми.

Клас Assistant описує основні функціональні можливості голосового асистента та успадковує клас QObject бібліотеки PyQt5 для належного функціонування в окремому потоці і передачі сигналів в головний потік. Даний клас має такі атрибути: публічний intents типу dict, публічний apps з типом dict, публічний mgr з типом WeatherManager, публічний stt з типом STT, публічний tts з типом TTS, публічний is_running типу bool. Клас Assistant налічує такі методи:

– hear(): публічний метод для безперервного очікування та розпізнання голосової команди від користувача;

– reply(request: str): публічний метод для попередньої обробки та надання фінальної відповіді;

– identify_intent(request: str): публічний метод для визначення вхідної команди;

– perform(command: str, option: str): публічний метод для обробки та виконання визначеної команди;

– speak(text: str): публічний метод для голосового відтворення відповіді;

– determine_app(input_n: str): публічний метод для визначення назви бажаної програми;

– search_apps(): публічний метод для пошуку всіх встановлених програм в операційній системі;

– stop(): публічний метод для безпечного завершення виконання роботи поточного потоку.

Клас STT описує функціонал розпізнавання мовлення та має такі атрибути: публічний asr_model, публічний total_buffer з типом float, публічний overhead_len з типом float, публічний tokens_per_chunk типу float та публічний mid_delay типу float. Даний клас містить такі методи:

– resample(audio: BytesIO): публічний метод який здійснює перетворення вхідного аудіо для отримання його аудіовмісту та метаданих у вигляді масиву NumPy;

– model(audio_16k: ndarray): публічний метод для отримання розподілу ймовірностей на кожній з можливих мовних одиниць;

– decode_predictions(logist_list): публічний метод для розшифровування передбачень, які надала навчена модель;

– transcribe(audio: BytesIO, state: list): публічний метод для розпізнавання тексту з вхідного аудіо інтерпретованого як об’єкт BytesIO.

Клас TTS бібліотеки ukrainian-tts описує функціонал синтезу мовлення та містить такі атрибути: публічний device типу str, публічний synthesizer типу Text2Speech та публічний xvectors типу dict. Даний клас має наступні методи:

– tts(text: str, voice: str, stress: str, output_fp: BytesIO(), speed: float): публічний метод який здійснює синтез вхідного тексту в голосовий аудіофайл інтерпретований в байтах;

– __setup_cache(cache_folder: str): приватний метод який дозволяє встановити папку-кеш для завантажування моделі синтезу, а також ініціалізує синтезатор з необхідними залежностями;

– __download(url: str, file_name: str): приватний метод який здійснює завантаження доступної моделі з посилання на репозиторій бібліотеки.

### 4.2.1 Модуль розпізнавання мовлення

На даний момент існує досить велика кількість натренованих моделей для розпізнавання українського мовлення, які є у відкритому доступі та надають можливість їх застосування у власних проектах, а саме: Wav2Vec2, DeepSpeech, Citrinet, Squeezeformer, VOSK та Whisper.

Проте вони значно відрізняються між собою як за точністю розпізнавання, так і за вимогою до обчислювальної потужності та ресурсів. Тому слід виділити найдоречнішу модель, яка буде задовольняти точністю розпізнавання українського мовлення та помірним використанням ресурсів персонального комп’ютера.

CER (Character Error Rate) та WER (Word Error Rate) – це метрики, що використовуються для оцінки якості розпізнавання мовлення штучними моделями.

CER визначає відношення кількості змінених, вставлених та вилучених символів до загальної кількості символів у тексті. CER дозволяє визначити точність розпізнавання кожного окремого символу в тексті.

WER визначає відношення кількості змінених, вставлених та вилучених слів до загальної кількості слів у тексті. WER відображає точність розпізнавання тексту в цілому, а не окремих символів.

Чим нижче значення CER або WER, тим краще якість розпізнавання. Однак, відповідні значення CER та WER можуть відрізнятися залежно від мовлення та його особливостей.

Також, слід зазначити, що розмір файлу моделі може впливати на швидкість розпізнавання та ресурсоємність. Зазвичай, більші розміри файлу означають загальну складність та широкий набір параметрів моделі, що може призводити до великої кількості обчислювальних операцій та значної вимоги до ресурсів.

Щодо швидкості розпізнавання, моделі з більшим розміром файлу можуть потребувати більше часу для обробки вхідних даних та здійснення прогнозів. Проте, моделі з меншим розміром файлу можуть працювати швидше та ефективніше, але можуть мати меншу точність розпізнавання.

Щодо ресурсоємності, то моделі з більшим розміром файлу можуть вимагати більшої кількості оперативної пам’яті та обчислювальної потужності. Це може стати проблемою, особливо при використанні на пристроях з обмеженими ресурсами.

В таблиці 4.1 наведено порівняння основних властивостей та метрик доступних моделей розпізнавання мовлення [8].

Таблиця 4.1 – Порівняння доступних моделей розпізнавання українського мовлення

| Назва моделі | WER | CER | Точність, % | Розмір |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| wav2vec2-xls-r-1b-uk-with-lm | 0,1807 | 0,0317 | 81,93 | 3,85 ГБ |
| wav2vec2-xls-r-300m-uk-with-wiki-lm | 0,2027 | 0,0365 | 79,73 | 1,26 ГБ |
| wav2vec2-xls-r-base-uk-with-small-lm | 0,4441 | 0,0975 | 55,59 | 378 МБ |
| DeepSpeech v0.5 | 0,7025 | 0,2009 | 29,75 | 700 МБ |

Продовження таблиці 4.1

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| stt_uk_citrinet_1024_gamma_0_25 | 0,0432 | 0,0094 | 95,68 | 567 МБ |
| stt_uk_citrinet_512_gamma_0_25 | 0,0746 | 0,016 | 92,54 | 143 МБ |
| stt_uk_squeezeformer_ctc_ml | 0,0591 | 0,0126 | 94,09 | 501 МБ |
| stt_uk_squeezeformer_ctc_xs | 0,1078 | 0,0229 | 89,22 | 36,8 МБ |
| VOSK v3 | 0,5325 | 0,3878 | 46,75 | 992 МБ |
| whisper-small-ukrainian | 0,2704 | 0,0565 | 72,96 | 967 МБ |
| whisper-large-uk-2 | 0,2482 | 0,055 | 75,18 | 6,17 ГБ |

Проаналізувавши доступні моделі розпізнавання українського мовлення, їхні метрики та властивості, для розробки власного голосового помічника було обрано модель stt-uk-squeezeformer-ctc-xs [9], яка проходила навчання на двох наборах даних: «Commo Voice 10» від Mozilla Foundation та «VOA Ukrainian dataset» (налічує близько 400 годин аудіо-даних). Ця модель показує досить високу точність та невеликий коефіцієнт помилок розпізнавання, враховуючи перевагу в найменшому розмірі серед розглянутих моделей.

Squeezeformer-CTC – це модель глибокого навчання для розпізнавання мови, заснована на гібридній згортково-трансформерній архітектурі з використанням алгоритму Connectionist Temporal Classification (CTC) для навчання без вчителя. Модель поєднує в собі дві архітектури: SqueezeNet та Transformer.

Архітектура SqueezeNet є легковаговою архітектурою згорткової нейронної мережі, яка була розроблена для зменшення кількості параметрів

у моделі без шкоди для її точності. SqueezeNet використовує блоки стиснення, які скорочують кількість фільтрів згортки у кожному шарі для зменшення кількості параметрів моделі.

Архітектура Transformer, у свою чергу, використовується для генерації вихідної послідовності, і вона була успішно застосована для задач машинного перекладу та розпізнавання мови. Вона складається з декількох шарів кодувальника та декодера, які використовують механізм уваги для обробки вхідних та вихідних послідовностей.

Squeezeformer-CTC об’єднує ці дві архітектури, використовуючи згорткову частину SqueezeNet для отримання ознак з аудіофайлів та трансформерну частину для генерації вихідної послідовності на основі одержаних ознак. На рисунку 4.3 наведено архітектуру моделі Squeezereformer-CTC [10].

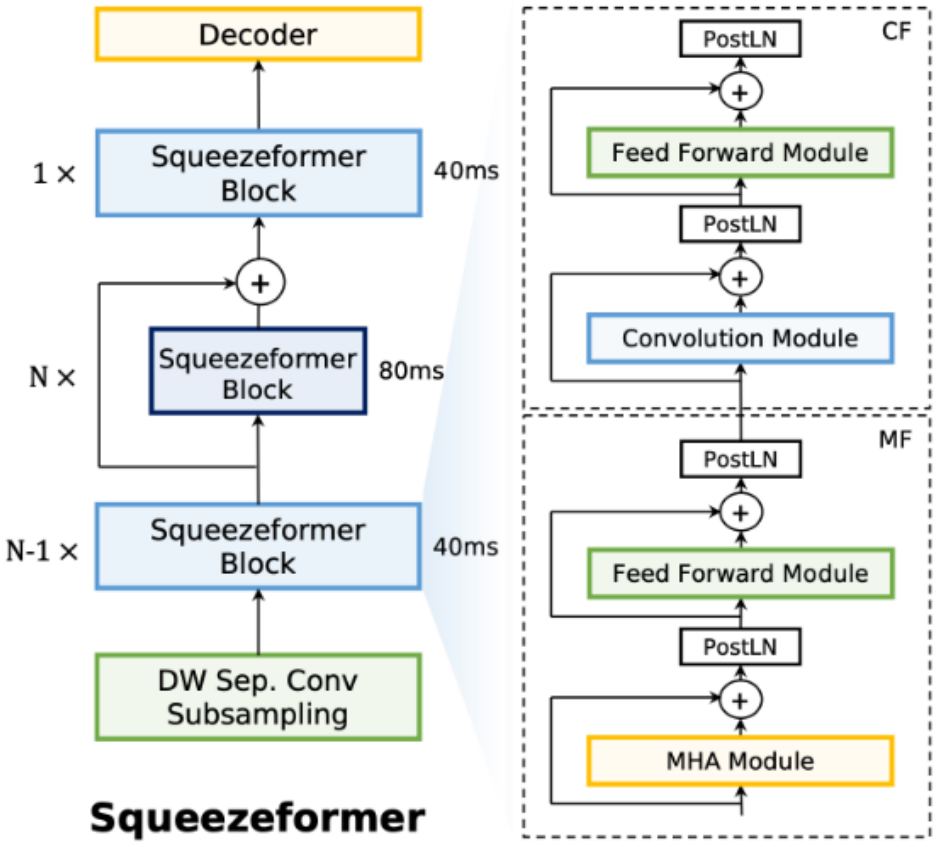


Рисунок 4.3 – Архітектура моделі Squeezeformer-CTC

Основні властивості архітектури Squeezeformer-CTC:

– використовує згорткові шари для отримання локальних ознак з аудіосигналу та трансформерні блоки з Multi-Head Attention для роботи з глобальною інформацією;

– використовує Feed Forward Module для обробки ознакових карт, отриманих після згорткових шарів у Feature Encoder та дозволяє зменшити їх розмірність до кількох сотень або тисяч ознак щоб отримати більш компактне представлення аудіосигналу;

– використовує CTC Loss для навчання без вчителя, що дозволяє уникнути проблеми з вирівнюванням вхідної та вихідної послідовностей;

– використовує Post-Layer Normalization (PostLN) після кожного шару уваги та повнозв'язного шару, щоб покращити якість розпізнавання та стабільність навчання;

– має відносно невелику кількість параметрів та швидко навчається на великих наборах даних.

Для навчання моделі Squeezeformer-CTC необхідні великі набори даних з аудіофайлами та відповідними транскрипціями. Модель навчається на основі пар аудіофайл-транскрипція, використовуючи алгоритм зворотного розповсюдження помилки.

Для використання обраної моделі знадобиться фреймворк для машинного навчання та бібліотеки для обробки аудіоданих. Одним із таких фреймворків є NeMo (Neural Modules), який надає високорівневий інтерфейс для створення та навчання моделей глибокого навчання. Фреймворк був розроблений компанією Nvidia, щоб допомогти розробникам швидко створювати та навчати складні моделі для різних завдань обробки природної мови.

Ключові особливості фреймворку Nvidia NeMo, які слід виділити:

– гнучкість: фреймворк дозволяє створювати та навчати моделі різних архітектур та компонентів, таких як конволюційні нейронні мережі (CNN), рекурентні нейронні мережі (RNN), трансформери та інші;

– простота використання: фреймворк має зручний інтерфейс та API, що дозволяє швидко створювати та навчати моделі;

– розширені можливості: фреймворк надає безліч можливостей, таких як автоматичне масштабування моделей для роботи з великими даними, підтримка розподіленого навчання, а також інтеграція з іншими бібліотеками глибокого навчання, такими як PyTorch і TensorFlow;

– підтримка різних форматів даних: фреймворк підтримує різні формати даних для розпізнавання мовлення, включаючи аудіофайли, транскрипції та мовні моделі.

– попередньо навчені моделі: NeMo надає широкий набір попередньо навчених моделей для розпізнавання мовлення різними мовами.

На рисунку 4.4 зображено фрагмент коду головного методу transcribe класу STT.

```
def transcribe(self, audio, state):
    if state is None:
        state = [np.array([], dtype=np.float32), []]

    audio_16k = self.resample(audio)
    state[0] = np.concatenate([state[0], audio_16k])

    while (len(state[0]) > self.total_buffer):
        buffer = state[0][:self.total_buffer]
        state[0] = state[0][self.total_buffer - self.overhead_len:]
        logits = self.model(buffer)
        state[1].append(logits)

    if len(state[1]) == 0:
        text = ""
    else:
        text = self.decode_predictions(state[1])

    return text, state
```

Рисунок 4.4 – Фрагмент коду методу transcribe класу STT

Даний метод транскрибує вхідне аудіо за допомогою обраної навченої моделі. Спочатку метод transcribe приймає аудіо інтерпретоване в байтах і

необов’язковий аргумент state, який є списком, що містить два елементи: масив numpy, що представляє поточний аудіобуфер, і список попередніх прогнозів моделі. Якщо аргумент state поступає на вхід як None, то він ініціалізується порожнім буфером і порожнім списком.

Далі відбувається конвертування метаданих вхідного аудіосигналу (дискретизацію 16 кГц і одноканальний режим) за допомогою методу resample для подальшого об’єднання цього аудіо в поточний буфера змінної state.

Потім циклом виконується проходження буферу методом ковзного вікна, пропускаючи його через попередньо навчену модель та додаючи отримані прогнози (logits) до списку в state.

Після обробки всього буферу метод декодує передбачення в текст за допомогою методу decode_predictions і повертає отриманий текстовий результат. Якщо модель не надала жодного прогнозу, то метод повертає порожній рядок.

Для захоплення вхідного аудіо з мікрофону, яке надходить від користувача, було використано бібліотеку SpeechRecognition. Вона містить вбудований функціонал, який допомагає з легкістю отримувати вхідний аудіопотік з мікрофону в реальному часі. На рисунку 4.5 наведено фрагмент коду методу hear з класу Assistant.

```
def hear(self):
    r = sr.Recognizer()
    if sr.Microphone.list_working_microphones():
        try:
            with sr.Microphone(sample_rate=16000) as source:
                r.adjust_for_ambient_noise(source, duration=1)
                while self.is_running:
                    try:
                        audio = r.listen(source, timeout=1)
                    except sr.WaitTimeoutError:
                        pass
                    else:
                        data = BytesIO(audio.get_wav_data())
                        text, _ = self.stt.transcribe(data, None)
                        if text and text.startswith(ALIAS):
                            self.change_img.emit(True)
                            self.reply(request=text)
                            self.change_img.emit(False)
        except OSError:
            self.off_mic.emit()
    else:
        self.no_mic.emit()
```

Рисунок 4.5 – Фрагмент коду методу hear з класу Assistant

Метод hear слугує вхідною точкою під час отримання віртуальним асистентом голосового запиту від користувача. Спочатку ініціалізується об’єкт типу Recognizer для подальшого використання потрібного функціоналу бібліотеки SpeechRecognition.

Далі програма перевіряє наявність доступного мікрофону в системі та за необхідності обробляє помилку в разі його відсутності. Клас Microphone визначає контекстний менеджер, який використовується для автоматичного відкриття та закриття доступу до мікрофона. Після того як знайдено активний мікрофон, відбувається секундне калібрування та пристосування до навколишніх шумів.

Створений Recognizer починає зчитувати вхідне аудіо в реальному часі для розпізнавання вимовлених слів методом transcribe, попередньо інтерпретувавши вхідний аудіосигнал в байтовий об’єкт. Якщо від користувача пролунало звернення до асистента, вказавши одне з його ключових імен, то дана текстова команда розпочне подальшу обробку методом reply з класу Assistant.

### 4.2.2 Модуль обробки та аналізу голосових команд

Для представлення бази знань голосового асистента було створено файл «knowledge» з форматом JSON. JSON (JavaScript Object Notation) – це формат даних, що використовується для передачі та зберігання структурованих даних у зручній для читання та запису формі. JSON-файли часто використовуються для обміну даними між клієнтськими та серверними програмами в Інтернеті.

JSON використовує простий і зрозумілий синтаксис, який було засновано на базі JavaScript та подає дані у вигляді пар «ключ-значення», які об’єднуються в об’єкти. Формат дозволяє представляти складні структури даних, включаючи об’єкти, списки та словники. Це робить JSON зручним для використання при передачі даних між різними системами та програмами, адже він не прив’язаний до конкретної мови програмування.

На рисунку 4.6 наведено фрагмент даних з файлу «knowledge.json».

```
"time": {
    "patterns": [
        "яка година",
        "котра година",
        "скільки годин",
        "час"
    ],
    "extra": ["скажи", "назви", "будь ласка", "поточний", "зараз"],
    "regex": [],
    "responses": [
        "поточний час становить",
        "на годиннику відображається",
        ""
    ],
    "warnings": []
},
"weather": {
    "patterns": [],
    "extra": ["зараз"],
    "regex": [
        "погод(?:а|у) (?:у|в) (?:населеному пункт|сел|селищ|міст)[\\S]* (?P<option>[а-щьюяїієґ'\\s\\-]+)$"
    ],
    "responses": [],
    "warnings": [
        "не можу знайти метеорологічну інформацію населеного пункту",
        "виникла помилка під час пошуку погодних даних",
        "погодна інформація по даному населеному пункту відсутня"
    ]
},
```


Рисунок 4.6 – Фрагмент даних бази знань голосового асистента

Файл бази знань голосового помічника містить структуроване подання команд, найменувань програм та відповідей на незрозумілі запити. Кожна команда має власний шаблон або набір шаблонів, ключові слова, готові відповіді та попередження.

Для обробки та ідентифікації відповідної команди від користувача і отримання ключових текстових ознак, було використано бібліотеку FuzzyWuzzy та регулярні вирази.

FuzzyWuzzy – це бібліотека для порівняння рядків, написана мовою Python. Вона надає ряд функцій для обчислення метрик подібності рядків, використовуючи різні методи, включаючи відстань Левенштейна. За допомогою FuzzyWuzzy можна легко порівнювати рядки, які можуть містити текстові помилки або різні варіації написання.

Відстань Левенштейна, також відома як відстань редагування, є метрикою, яка використовується для вимірювання різниці між двома рядками. Вона вимірює мінімальну кількість операцій вставки, видалення або заміни символів, які необхідно виконати, щоб перетворити один рядок на інший.

Кожна операція вставки, видалення або заміни символів має властивість вартості, які зазвичай є майже однаковими. Таким чином, для обчислення відстані Левенштейна між двома рядками необхідно знайти шлях з мінімальною вартістю від одного рядка до іншого, використовуючи тільки ці три операції.

Ця метрика широко використовується в комп’ютерній науці, особливо в галузі комп’ютерної лінгвістики та нечіткої логіки. Вона може бути використана для визначення подібності текстів, виправлення помилок введення, розпізнавання мови та багатьох інших завдань.

Регулярні вирази (Regular Expressions, скорочено RegExp або regex) – це послідовність символів, які використовуються для пошуку та обробки текстових даних. Вони є потужним інструментом для роботи з текстом і дозволяють виконати широкий спектр завдань, таких як пошук, заміна,

валідація та отримання інформації з тексту. Регулярні вирази складаються з спеціальних символів, які позначають відповідні символьні набори та класи та певні патерни, які потрібно знайти в тексті.

Ця методика часто використовується у багатьох мовах програмування, таких як Python, JavaScript чи Java, а також у текстових редакторах та інструментах командного рядка для обробки тексту.

Однак використання регулярних виразів для визначення команд віртуальним асистентом є виснажливим завданням та потребує правильного налаштування і тестування. Тому для розробки складних широко функціональних голосових асистентів краще використовувати більш просунуті методи обробки природної мови та навчені штучні моделі розпізнавання текстових даних, щоб поліпшити точність і загальну якість роботи програми.

На рисунку 4.7 наведено фрагмент коду методу identify_intent з класу Assistant. Даний метод виконує пошук відповідної текстової команди за встановленими шаблонами бази знань.

```
def identify_intent(self, request):
    for intent in sorted(self.intents, key=lambda k: self.intents[k]['regex'],
                         reverse=True):
        clean_req = request
        if extra_words := self.intents[intent]['extra']:
            for word in extra_words:
                clean_req = re.sub(fr"\b{word}\b\s?", '', clean_req)
            clean_req = clean_req.rstrip()
        if regex_patterns := self.intents[intent]['regex']:
            for pattern in regex_patterns:
                if res := re.search(fr"{pattern}", clean_req):
                    return intent, res.group('option')
        if patterns := self.intents[intent]['patterns']:
            for pattern in patterns:
                if fuzz.ratio(clean_req, pattern) >= 95:
                    return intent, None
    return None, None
```

Рисунок 4.7 – Фрагмент коду методу identify_intent

### 4.2.3 Модуль синтезу мовлення

Для розробки синтезу мовлення голосового асистента, було задіяно бібліотеку Ukrainian-TTS версії 5.0, яка використовує функціональні можливості синтезу ESPnet.

До головних особливостей Ukrainian-TTS можна віднести наступне:

– повноцінне функціонування без доступу до мережі Інтернет;

– вибір голосу синтезу з чотирьох доступних (два чоловічі та два жіночі);

– автоматичне розставлення наголосів із пріоритетною чергою (власне, словникове та з використанням моделі);

– керування швидкістю мовлення;

– підтримка функціонування на різних операційних системах, а саме Windows, Mac (x86/M1) та Linux (x86/ARM);

– виведення на мобільних пристроях (моделі виведення через espnet_onnx без очисників).

Розробник бібліотеки для її використання надає власну модель синтезу мовлення з архітектурою VITS, яка навчалася протягом 72 годин на наборі даних «Open Source Ukrainian Text-to-Speech datasets» (містить близько 35 годин записаних аудіо даних) в 352800 кроків за рахунок обчислювальної потужності відеокарти Nvidia RTX 3090.

ESPnet – це інструмент для дослідження та розробки мовних технологій, включаючи розпізнавання мовлення, машинний переклад та синтез мовлення. ESPnet був розроблений університетом Кіото (Kyoto University) та іншими організаціями, такими як Sony Corporation і Nara Institute of Science and Technology, в Японії.

Розробка ESPnet почалася у 2017 році та вже через рік він став доступним для загального використання. З тих пір фреймворк отримав широку популярність в галузі обробки мовлення та машинного навчання, і використовується для багатьох дослідницьких та комерційних проектів.

ESPnet підтримує різні архітектури нейронних мереж, які можуть бути використані для технології TTS, включаючи Tacotron2, Transformer-TTS, FastSpeech2, VITS та JETS. Кожна з цих моделей має свої переваги та недоліки, і підходить для різних типів програм. Фреймворк також містить набір інструментів, що дозволяють працювати з різними типами та форматами, зокрема з аудіо та текстовими даними.

В разі відсутності необхідних файлів для належного функціонування бібліотеки Ukrainian-TTS, вона автоматично завантажує до кеш-папки модель, конфігурацію та файл «кросвекторів» (xvectors). Після цього увесь процес синтезу мовлення відбувається за рахунок ресурсів комп’ютера на якому використовується голосовий помічник.

У ESPNet, xvectors утворюються шляхом обробки аудіосигналів за допомогою глибокої нейронної мережі, яка отримує характеристики мови з вхідного аудіо і перетворює їх на векторне уявлення. Потім xvectors використовуються для ідентифікації дикторів та інших завдань обробки мовлення, а також можуть включати широкий набір функцій, що характеризують мовлення, а саме тональність, висота голосу, швидкість мови, акцент і т. д.

VITS (Conditional Variation Autoencoder with Adversarial Learning for End-to-End Text-to-Speech) – це одна з найсучасніших моделей синтезу мовлення та була запропонована в 2020 році.

Модель VITS базується на архітектурі умовного варіаційного автокодувальника (CVAE), яка є типом нейронної мережі, що може навчитися генерувати вибірки даних, які залежать від певних вхідних даних. У випадку TTS на вході є текстова розшифровка, а на виході – відповідний мовний сигнал. Модель VITS навчається наскрізно, тобто вона приймає необроблений текст як вхідні дані та створює мовний сигнал як вихідні, не покладаючись на жодні проміжні функції.

Однією з ключових особливостей моделі VITS є використання змагального навчання. Це передбачає навчання окремої мережі

дискримінатора, яка намагається відрізнити згенерований мовний сигнал від реальних зразків мовлення, тоді як генератор (CVAE) намагається створювати мовні сигнали, які неможливо відрізнити від справжньої мови. Навчаючи генератор «обманювати» дискримінатор, модель VITS може навчитися створювати високоякісні мовні сигнали, які дуже схожі на реальну мову [11].

Іншою важливою особливістю моделі VITS є використання нею схеми навчання «вчитель-учень». У цій схемі окрема модель «вчителя» використовується для забезпечення додаткового контролю під час навчання, направляючи автокодувальник на створення більш точних мовних сигналів. Модель викладача – це окрема модель TTS, яка вже навчена на великому наборі даних і надає додаткову інформацію про те, як має звучати згенероване мовлення.

На рисунку 4.8 зображено схему методики, яка використовується для навчання моделі VITS [11].

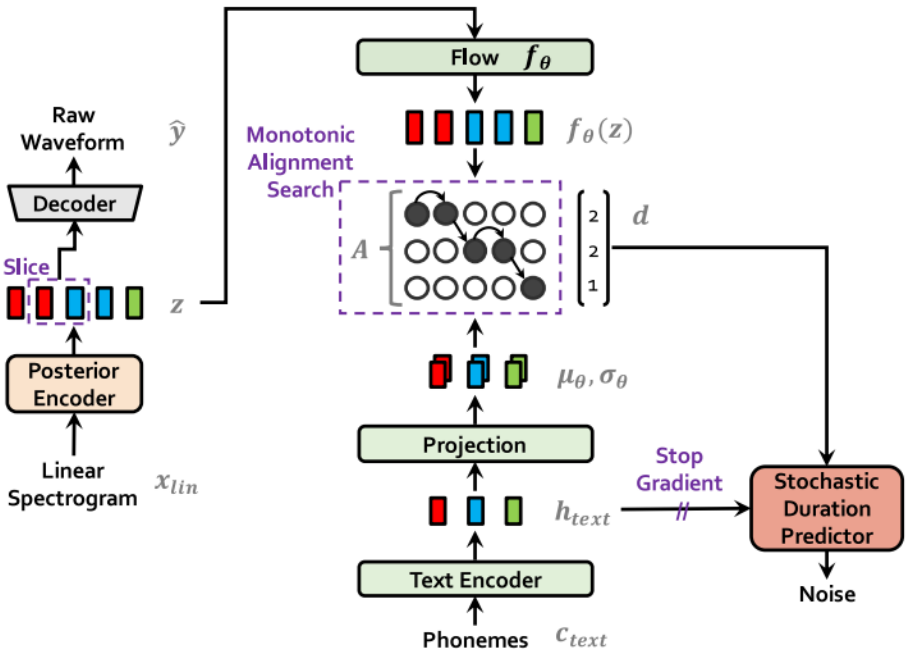


Рисунок 4.8 – Схема методики навчання моделі VITS

Загалом, модель VITS є значним прогресом у сфері TTS, оскільки вона здатна створювати дуже реалістичні мовні сигнали, які інколи важко відрізнити від справжнього мовлення.

Для реалізації синтезу мовлення голосового асистента, було створено окремий метод speak в класі Assistant, використовуючи доступні можливості та модель вищезгаданої бібліотеки Ukrainian-TTS. На рисунку 4.9 наведено фрагмент коду методу speak.

```
def speak(self, text):
    voice = BytesIO()
    _, _, _ = self.tts.tts(text, Voices.Tetiana.value,
                           Stress.Dictionary.value,
                           output_fp=voice, speed=0.9)
    pg.mixer.music.load(voice, "wav")
    pg.mixer.music.play()
    while pg.mixer.music.get_busy() and self.is_running:
        pg.time.Clock().tick(100)
    pg.mixer.music.stop()
```

Рисунок 4.9 – Фрагмент коду методу speak з класу Assistant

Спочатку модель перетворює вхідний текст в голосовий аудіо-фрагмент і записує його в байтовий об’єкт який тимчасово зберігається в оперативній пам’яті. Далі це аудіо відтворюється використовуючи можливості модуля Mixer бібліотеки PyGame.

Mixer – це модуль, що надає зручний інтерфейс для роботи зі звуком в Python. Він надає можливість керувати гучністю звуку, налаштовувати параметри звукових ефектів та забезпечувати багатопоточність під час програвання звуку. Це робить його дуже зручним та простим інструментом для створення програм, які використовують звукове відтворення і аудіо-ефекти.

# 5 МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ ПРИКЛАДНОЇ ПРОГРАМИ

В ході виконання кваліфікаційної роботи, було створено власну прикладну програму віртуального голосового асистента, використовуючи описані методи та технології.

Розроблений варіант здатний вирішити та мінімізувати основні проблеми сучасних віртуальних помічників під керуванням операційної системи Windows, та поєднує в собі такі переваги:

– має повноцінну українську локалізацію та підтримку україномовних користувачів;

– не зберігає вхідну інформацію запитів користувачів, тим самим не порушуючи проблему конфіденційності та безпеки даних;

– здатен функціонувати без постійного підключення до мережі Інтернет.

Для використання голосового асистенту, користувач запускає розроблену програму та за бажанням мінімізує основне вікно, залишивши працювати її у фоновому режимі.

Програма має доволі простий та зрозумілий графічний інтерфейс, який відображає тільки два стани активності голосового асистента: «очікування команди» та «виконання команди», що відповідно зазначено на рисунках 5.1 та 5.2.

Якщо операційна система не має активного мікрофону чи голосовий помічник не може отримати до нього доступ, буде надано попередження, в разі вирішення якого буде можливість подальшого використання програми (рисунок 5.3).

Голосовий асистент розпочинає працювати в режимі очікування ключового слова, а саме імені «Марічка», після якого повинен слідувати бажаний голосовий запит. Як тільки голосовий асистент отримує звернення від користувача, відбувається ідентифікація та аналіз відповідної команди.

Якщо програма змогла визначити команду, вона виконує відповідну дію на дану вимогу та за необхідності надає голосову відповідь. В іншому випадку, якщо віртуальний асистент не зміг зрозуміти намір користувача, або виникла певна помилка під час цього процесу, то буде надано голосову відповідь зазначивши характер цієї проблеми.

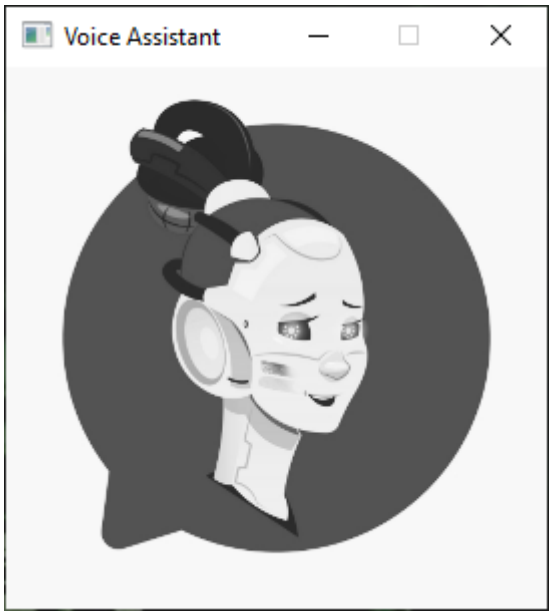


Рисунок 5.1 – Екранна форма програми в режимі «очікування команди»

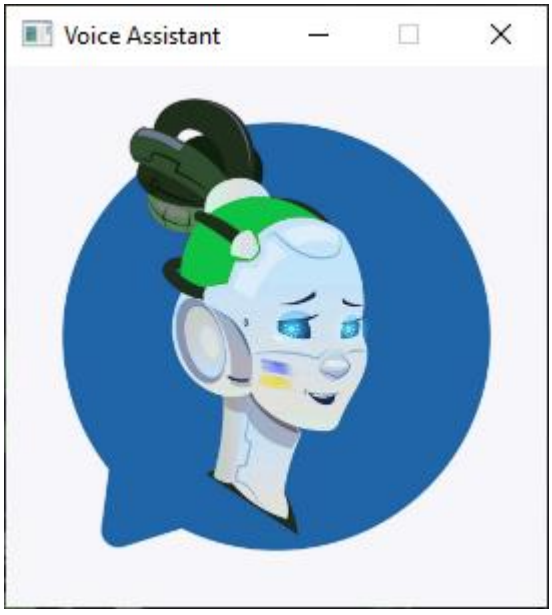


Рисунок 5.2 – Екранна форма програми в режимі «виконання команди»

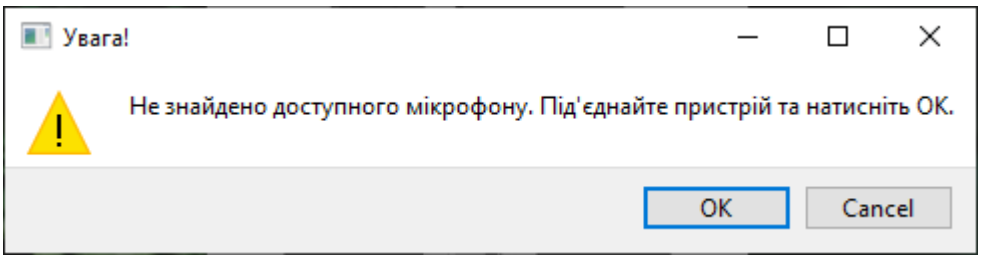


Рисунок 5.3 – Попередження про відсутність активного мікрофону

Для зручного та ефективного користування комп’ютером, розроблений голосовий асистент забезпечує наступні функціональні можливості:

– здійснює розпізнавання в реальному часі голосових команд від користувача;

– називає поточний час;

– знаходить погодну інформацію вказаного населеного пункту та надає голосову відповідь;

– знаходить і відкриває програму вказану користувачем, що встановлена на комп’ютері;

– виконує пошук користувацького запиту в браузері;

– зберігає знімок екрану та відображає його за проханням користувача.

В якості прикладу, на рисунках 5.4 та 5.5 відповідно, наведено результат виконання голосового запиту «Марічка, запусти налаштування будь ласка» та консольне відображення чіткого розуміння голосовим асистентом даної команди.

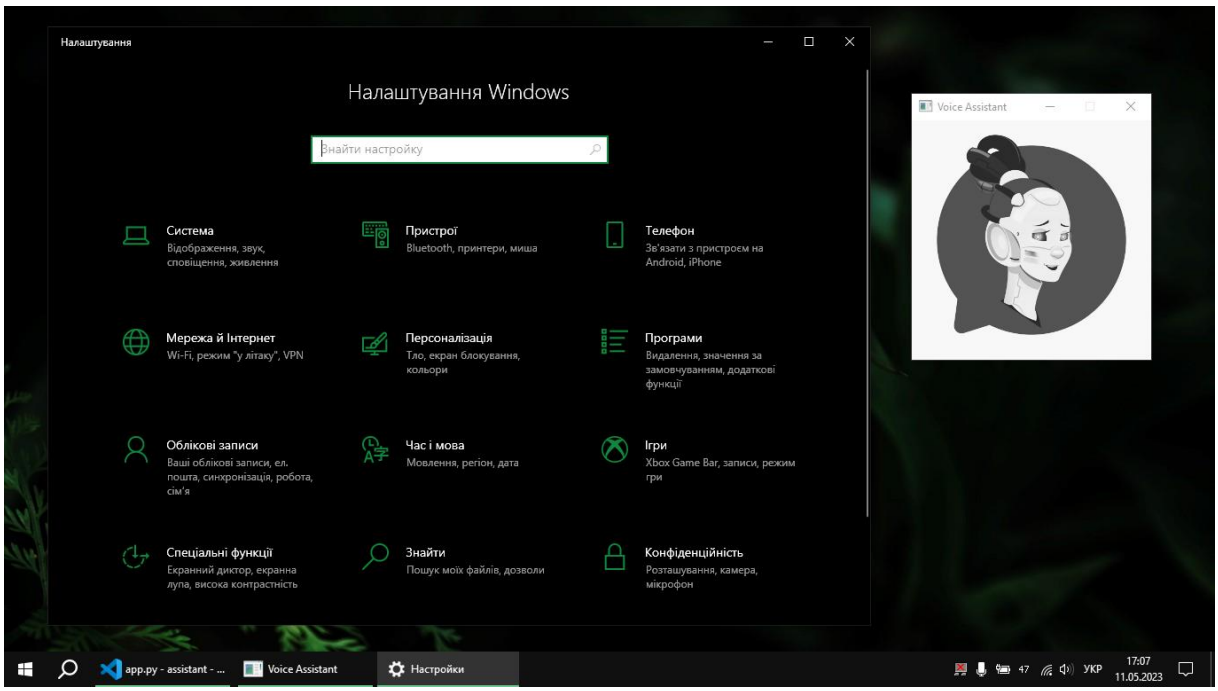


Рисунок 5.4 – Результат виконання голосового запиту від користувача

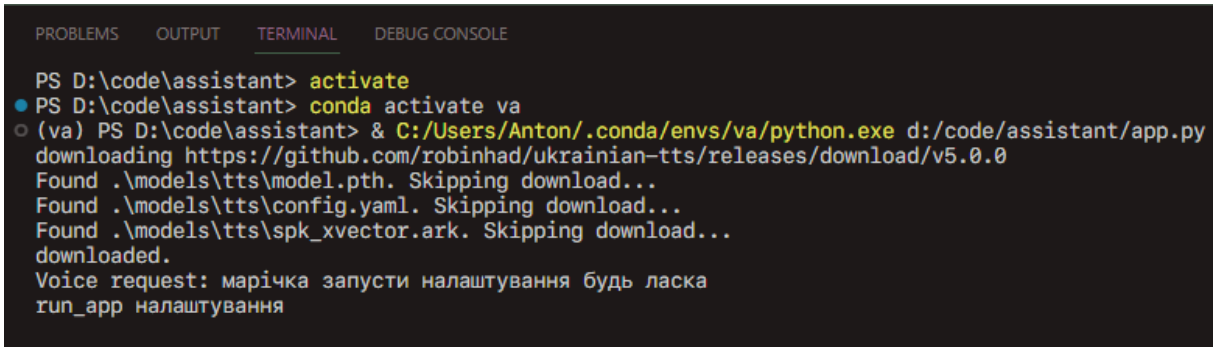


Рисунок 5.5 – Консольне відображення чіткого розуміння команди

На рисунку 5.6 наведено UML діаграму прецедентів (Use Case), яка відображає сценарії використання функціоналу розробленої прикладної програми.

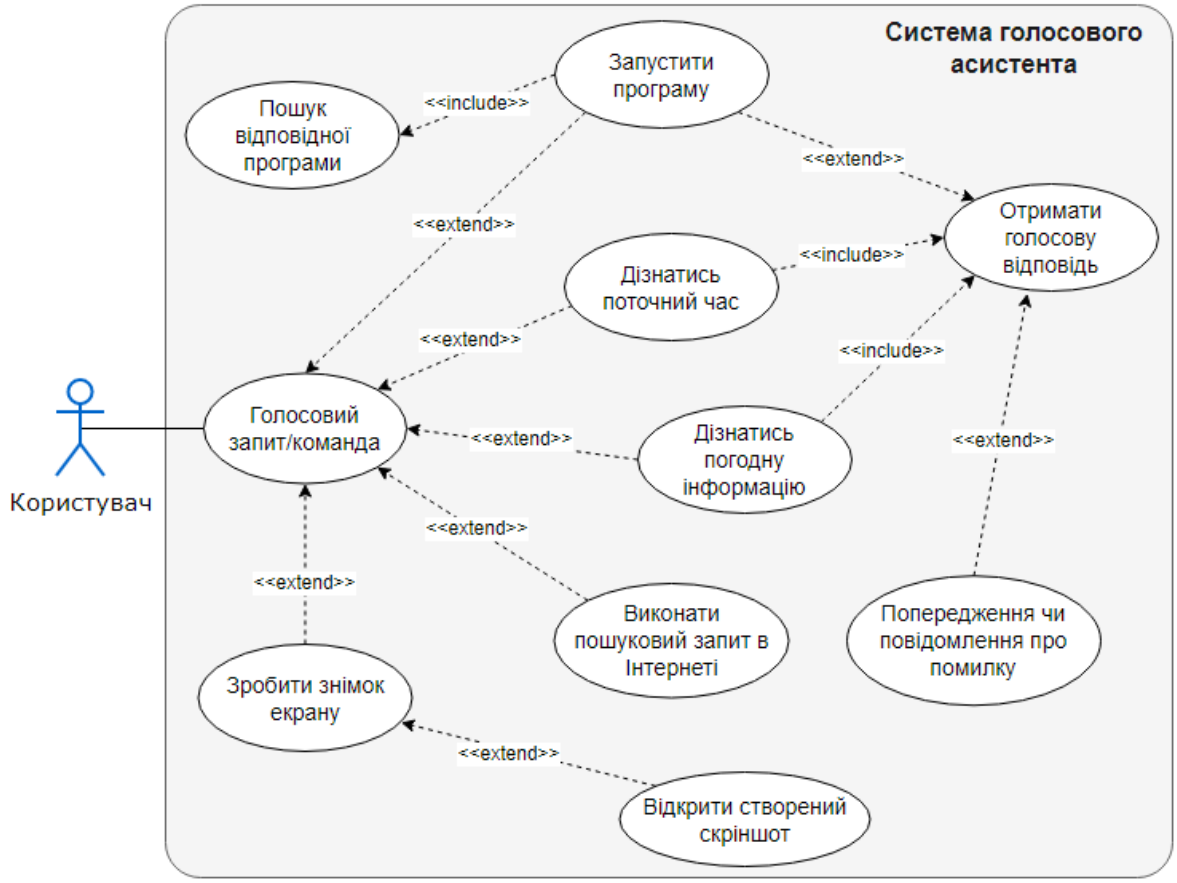


Рисунок 5.6 – Діаграма прецедентів використання голосового асистента

Розробка, налагодження та попереднє тестування створеного голосового асистента виконувалось на комп'ютері з наступними характеристиками: Windows 10 64-бітна версія, процесор Intel Core i3-8100 3,6 ГГц з чотирма ядрами, 16 ГБ оперативної пам'яті, відеокарта Nvidia GTX 1050 Ti.

Також тестування та перевірку належного функціонування програми було здійснено на незначному за потужністю ноутбуці з такими характеристиками: Windows 10 64-бітна версія, процесор Intel Core i3-6100U 2,3 ГГц з двома ядрами, 8 ГБ оперативної пам'яті та відеокарта Nvidia 920MX.

Розроблений голосовий асистент показав задовільні результати за точністю та швидкістю розпізнавання і синтезу мовлення на обох системах.

# ВИСНОВКИ

Кваліфікаційна робота магістерського рівня на тему «Програмний модуль голосового асистента для синтезу та розпізнавання мовлення» є важливим та актуальним дослідженням у сфері штучного інтелекту і технологій голосового управління.

Головною метою даної роботи була розробка голосового інтерфейсу для зручної та інтерактивної взаємодії україномовних користувачів з власним комп'ютером під керуванням операційної системи Windows.

Під час виконання кваліфікаційної роботи було проведено аналіз щодо предметної області та розглянуто голосових асистентів як інноваційний технологічний продукт, що піднімає способи користування сучасними пристроями на новий рівень.

Було досліджено технічні та функціональні особливості наявних голосових асистентів, виділено їхні можливості, обмеження та недоліки, здійснено аналіз використання таких систем в різних галузях, та в свою чергу розглянуто основні методи та підходи для їх розробки.

Особливу увагу було приділено питанням конфіденційності та захисту персональних даних користувачів, відсутності української локалізації та офіційної підтримки україномовних користувачів, а також наявності постійного підключення до мережі Інтернет під час користування такою системою.

В результаті цього було запропоновано рішення та розроблено власну прикладну програму голосового помічника для подолання головних проблем сучасних голосових асистентів. Створений віртуальний асистент показав достатньо чудові показники точності, ефективності та швидкодії під час розпізнавання та синтезу мовлення, використовуючи добре підібрані методи та технології.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Capgemini. How organizations and consumers are embracing voice and chat assistants, 2019. URL: https://www.capgemini.com/wp-content/uploads/2019/09/Report-%E2%80%93-Conversational-Interfaces_Web-Final.pdf (дата звернення: 15.04.2023)

2. Tankovska H. Number of digital voice assistants in use worldwide 2019–2024 (in billions), 2020. URL: https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/ (дата звернення: 16.04.2023)

3. Zwakman DS et al. Voice usability scale: measuring the user experience with voice assistants. In: 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS). IEEE; 2020.

4. IBM Shoebox. URL: https://en.wikipedia.org/wiki/IBM_Shoebox (дата звернення: 16.04.2023)

5. Voice Assistant Use Cases: Business Implementations of VUIs in 2023. URL: https://masterofcode.com/blog/voice-assistant-use-cases-business-implementations-of-vuis-in-2021 (дата звернення: 17.04.2023)

6. Amazon wages secret war on Americans' privacy, documents show, 2021. URL: https://www.reuters.com/investigates/special-report/amazon-privacy-lobbying (дата звернення: 18.04.2023)

7. Voice Assistant Application for Avoiding Sedentarism in Elderly People Based on IoT Technologies, 2021. URL: https://doi.org/10.3390/electronics10080980 (дата звернення: 27.04.2023)

8. Speech Recognition for Ukrainian. URL: https://github.com/egorsmkv/speech-recognition-uk (дата звернення 05.05.2023)

9. Squeezeformer-CTC XS (uk-UA). URL: https://huggingface.co/theodotus/stt_uk_squeezeformer_ctc_xs (дата звернення: 07.05.2023)

10. Sehoon Kim, Amir Gholami, Albert Shaw, Nicholas Lee, Karttikeya Mangalam, Jitendra Malik, Michael W. Mahoney, Kurt Keutzer. Squeezeformer: An Efficient Transformer for Automatic Speech Recognition, 2022.

11. Jaehyeon Kim, Jungil Kong, Juhee Son. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech, 2021.

# ДОДАТОК А

Відомість кваліфікаційної роботи

| Позначення | Найменування | Дод. відомості |
|---|---|---|
| | Текстові документи | |
| **1.** | Пояснювальна записка | 62 с. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | Інші документи | |
| | | |
| **2.** | Презентаційні матеріали | 16 плакатів |
| | | |
| | | |
| | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| . | . | Прізвище та ініціали. | Підп. | Дата | |
| Розробив | | Кулик А. О. | | | Програмний модуль голосового асистента для синтезу та розпізнавання мовлення |
| Перевірив | | Сніжко Д. В. | | | |
| Н.контр. | | Малєєва І.А. | | | |
| Затв. | | Філатов В.О. | | | |

| Шифр групи | Код напр./спец. |
|---|---|
| СШІм-21-2 | 122 |
| ХНУРЕ кафедра ШІ | |