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)

Голосовий асистент користувача на базі нейромережевих технологій

(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Липенко А.В.
(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 проф. Єрохін А.Л.
(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 Липенку Артему Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Голосовий асистент користувача на базі нейромережевих технологій

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, відкриті джерела в мережі Інтернет, відкриті набори даних, середовище розробки PyCharmCommunity 2022.2

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Постановка задачі .

3) Комп'ютерна система

4) Програмування та тестування

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вивчення предметної області дослідження	05.04.2023	Виконано
2	Дослідження літератури	08.04.2023	Виконано
3	Вивчення проблеми, що потребує рішення	11.04.2023	Виконано
4	Формування постановки задачі дослідження	12.04.2023	Виконано
5	Аналіз існуючих підходів	15.04.2023	Виконано
6	Концептуальне проектування методу рішення задачі	17.04.2023	Виконано
7	Проектування компонентів ПЗ	20.04.2023	Виконано
8	Розробка компонентів ПЗ	23.04.2023	Виконано
9	Тестування працездатності розробленого ПЗ	28.04.2023	Виконано
10	Оформлення пояснювальної записки	05.05.2023	Виконано
11	Попередній захист	12.05.2023	Виконано
12	Захист перед ЕК	18.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент  _____
(підпис)

Керівник роботи _____ проф. Єрохін А.Л.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 73 с., 12 рис., 2 табл., 2 дод., 41 джерело.

ВЕКТОРИЗАЦІЯ, ВИБІРКА, ГОЛОСОВИЙ АССИСТЕНТ, НЕЙРОННА МЕРЕЖА, НЕЙРОМЕРЕЖЕВІ ТЕХНОЛОГІІ, ПОПЕРЕДНЯ ОБРОБКА.

Об'єкт дослідження – дослідження та визначення підходящих нейромережевих технологій.

Предмет дослідження – пошук методів використання нейронних мереж для створення голосового помічника.

Мета роботи – дослідження нейронних мереж та створення на їх базі голосового асистента.

Методи дослідження – аналіз технічної літератури з області нейромережевих технологій, експериментальний підбір нейронних мереж, програмна реалізація та проведення комп’ютерного експерименту.

Проведено теоретичний аналіз різних нейронних мереж та їх архітектур, методів оптимізації та інших параметрів. Проведення теоретичних та практичних дослідів використання нейронних мереж для створення голосового помічника. Було виділено основні недоліки існуючих підходів та запропоновано простий спосіб побудови програми на базі нейронних мереж. На основі отриманих результатів розроблено прототипну модель, за допомогою якої можна легко користуватися будь-яким пристроєм та доповнювати модель новими технологіями.

ABSTRACT

Explanatory note: 73 p., 12 fig., 2 tabl., 2 ann., 41 sources.

NEURAL NETWORK TECHNOLOGIES, PRE-PROCESSING, SAMPLING, VECTORIZATION, VOICE ASSISTANT.

The object of follow-up is the follow-up of the designation of suitable neuro-branch technologies.

The subject of the study is the search for methods of finding neural networks for the creation of a voice helper.

Meta robotics – continuation of neural networks on the basis of the voice assistant.

Research methods – analysis of technical literature from the field of neuronetwork technologies, experimental selection of neural networks, software implementation and computer experiment.

A theoretical analysis of various neural networks and architectures, optimization methods and other parameters was carried out. Carrying out theoretical and practical studies of the selection of neural networks for the creation of a voice assistant. We have seen the main shortfalls of the essential approaches and proposed a simple way to encourage programs based on neural networks. On the basis of taking the results, a prototype model was broken up, for which it is possible to easily build up, be it some kind of extension, and supplement the model with new technologies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ШНМ – штучні нейронні мережі;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – прикладний програмний інтерфейс;

CNN – Convolutional neural network – згорткова нейронні мережа;

DNN – Deep neural network – глибинна нейронна мережа;

GNN – Graph neural network – графова нейронна мережа;

LSTM – Long short-term memory – довга короткочасна пам'ять;

NLP – Natural Language Processing – нейролінгвістичне програмування, кодування психіки;

RNN – Recurrent neural network – рекурентна нейронна мережа.

ВСТУП

Штучний інтелект (ШІ) є однією з найбільш важливих та швидкозростаючих галузей технологій в сучасному світі. Він постійно розвивається та вдосконалюється, забезпечуючи нові можливості для бізнесу, медицини, науки та багатьох інших галузей.

Його основна – це створення програм та систем, які здатні імітувати розум людини та здійснювати різні завдання. Для цього використовуються алгоритми та моделі, що забезпечують машинам здатність до самостійного навчання та покращення своїх функцій.

Штучний інтелект може бути використаний для вирішення різних завдань та проблем, що стикаються люди. Він допомагає підвищувати ефективність та якість життя людей, що є важливим для розвитку суспільства в цілому. Крім того, ШІ може бути використаний для здійснення наукових досліджень та розробок нових технологій, що може привести до нових відкриттів та винаходів.

Щодо конкретних використань можна навести декілька прикладів. ШІ може бути використаний для створення голосових помічників, автономних транспортних засобів, систем розпізнавання образів та багато іншого.

Розвиток штучного інтелекту змінює світ навколо нас та надає нові можливості для досягнення успіху та вирішення складних проблем. Він є однією з головних технологій майбутнього та далі буде продовжувати здивовувати своїми можливостями.

Штучний інтелект може допомогти людям у багатьох галузях життя, зокрема в науці, медицині, транспорті, бізнесі та особистому житті. Ось деякі з переваг, які він може надати людям:

– покращення ефективності та продуктивності: штучний інтелект може виконувати складні завдання швидше та точніше, ніж людина. Наприклад, він може автоматизувати процеси виробництва, обробки даних та аналізу великих обсягів інформації;

– поліпшення якості послуг: за допомогою штучного інтелекту розробка більш точних та персоналізованих продуктів та послуг стає простішою, забезпечуючи краще задоволення потреб клієнтів;

– виявлення та запобігання помилок: штучний інтелект може виявляти та усувати помилки швидше за людину, що зменшує ризики небезпеки та підвищує безпеку;

– зменшення витрат: штучний інтелект може допомогти зменшити витрати на працю та операції, оскільки він може автоматизувати багато рутинних завдань;

– вдосконалення медицини: штучний інтелект може допомогти у виявленні хвороб, аналізі медичних даних та розробці більш ефективних методів лікування;

– підвищення рівня безпеки: штучний інтелект може допомогти виявляти загрози безпеці та запобігати їм, наприклад, у сфері кібербезпеки.

Дослідження з вивчення штучного інтелекту розпочалися ще в середині XX століття. Професор Дартмутського коледжу, Джон Маккарті, в 1956 році сформулював поняття штучного інтелекту як науки, а відомий Тест Тьюринга став важливою віхою в наукових дослідженнях цієї області [31].

Звичайно, суперечки про те, чи можна вважати ШІ інтелектом повною мірою, до цього часу продовжуються. Але це не суперечить тому, що розумні системи, які навчаються, полегшують життя людини, хоча за рівнем розвитку пасуть задніх.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної галузі

Голосові асистенти користувача на базі нейромережевих технологій є однією з найбільш перспективних технологій в галузі штучного інтелекту. Вони дозволяють користувачам взаємодіяти з різними пристроями, використовуючи лише свій голос. Це забезпечує зручність та швидкість взаємодії, яка зростає важливістю в умовах швидкого розвитку технологій та зростаючої потреби в швидкому та ефективному доступі до інформації.

Одним з головних завдань розробки голосових асистентів на базі нейромереж є створення моделей, які здатні точно розпізнавати мову та інтелектуально обробляти інформацію, отриману від користувачів. Для досягнення цієї мети використовуються методи навчання машинного навчання, зокрема глибоке навчання, яке дозволяє створювати нейромережеві моделі, здатні до аналізу великої кількості даних та відповідності нових запитів.

Незважаючи на переваги голосових асистентів, деякі користувачі можуть стурбовуватися за приватність, яка може бути порушена в результаті збору та обробки персональних даних. Тому важливо розробляти технології, які забезпечують безпеку та захист конфіденційності даних користувачів [34].

У даному дослідженні я буду розглядати основні принципи роботи голосових асистентів на базі нейромереж та їх застосування у різних галузях, а також проведу дослідження питання приватності та безпеки при використанні цих технологій. Моя мета полягає в тому, щоб дослідити можливості використання голосових асистентів на базі нейромереж в різних сферах життя, включаючи медицину, освіту, торгівлю та інші.

Крім того, я проаналізую існуючі проблеми та виклики, пов'язані з використанням голосових асистентів, такі як недостатня точність

розпізнавання мови, складність взаємодії з користувачами, а також проблеми з безпекою та приватністю.

В результаті цього дослідження я сподіваюсь розкрити потенціал голосових асистентів на базі нейромереж та визначити можливі шляхи подальшого розвитку та вдосконалення цих технологій. Також я сподіваюсь знайти способи забезпечення безпеки та приватності даних користувачів, що дозволить розширити можливості використання голосових асистентів в різних галузях життя.

1.2 Огляд нейронних мереж

Нейронні мережі – це математичні моделі, що імітують роботу людського мозку. Вони складаються зі спеціальних структурних одиниць, які називають нейронами. Ці нейрони з'єднані між собою, утворюючи велику мережу, яка може виконувати складні завдання, такі як розпізнавання образів, розуміння мови та інші.

Нейронні мережі були розроблені в 1940-х роках, але їх популярність значно зросла в останні десятиліття завдяки поступовому збільшенню обчислювальної потужності та доступності великої кількості даних. Сьогодні нейронні мережі використовуються в багатьох сферах, таких як комп'ютерне зору, обробка природних мов, автономні системи керування, розпізнавання мови та багато інших [35].

Одним з видів нейронних мереж є глибокі нейронні мережі, які складаються з багатьох шарів нейронів. Ці мережі дозволяють досягти високої точності в розв'язанні складних задач, що раніше були недосяжні для комп'ютерів. Глибокі нейронні мережі застосовуються в багатьох задачах машинного навчання, таких як класифікація образів, розпізнавання мови та аналіз даних. Вони можуть навчатися на великих наборах даних, а потім застосовуватися для розв'язання нових задач [33].

Незважаючи на успіхи, які були досягнуті в галузі нейронних мереж, вони також мають свої виклики та обмеження. Наприклад, вони можуть бути дуже чутливі до шуму в даних та потребують великої кількості обчислювальних ресурсів для тренування та використання. Також, нейронні мережі можуть бути дуже складними для інтерпретації, тому важко розуміти, як саме вони приймають рішення. Проте, незважаючи на ці обмеження, вони є потужним інструментом, який знаходить застосування в багатьох галузях, включаючи голосових асистентів. Глибокі нейронні мережі можуть використовуватися для розпізнавання голосу та розуміння мовлення, що дозволяє голосовому асистентові взаємодіяти з користувачем за допомогою мовлення.

Одним з викликів при використанні нейронних мереж в голосових асистентах є потреба великої кількості даних для тренування. Чим більше дані, тим більш точніші результати можуть бути отримані в результаті тренування. Також, важливим фактором є розмір нейронної мережі, який повинен бути достатньо великим, щоб забезпечити високу точність, але при цьому не ставити питання про обчислювальні ресурси [36].

Огляд нейронних мереж демонструє, що вони є потужним інструментом для вирішення складних завдань. Глибокі нейронні мережі зокрема, є найбільш ефективним інструментом машинного навчання на сьогоднішній день. Вони знаходять застосування в багатьох галузях, включаючи голосові асистенти, і мають великий потенціал для подальшого розвитку технологій, які дозволяють користувачам взаємодіяти з комп'ютерами та іншими пристроями за допомогою мовлення.

Використання нейронних мереж для створення голосового асистента є одним із ключових елементів цієї технології. Нейромережі дозволяють асистентам розпізнавати та розуміти мову, генерувати голосові відповіді та виконувати інші завдання.

Основні кроки створення голосового асистента з використанням нейромереж включають в себе такі елементи:

– для того, щоб голосовий асистент міг розуміти користувача, необхідно мати механізм розпізнавання мови. Це може бути досягнуто за допомогою нейронних мереж, зокрема рекурентних нейронних мереж (RNN) або сверточних нейронних мереж (CNN), які здатні аналізувати голосові сигнали та перетворювати їх на текст;

– після розпізнавання мови голосовий асистент повинен розуміти смисл команди користувача. Це можна досягти за допомогою методів обробки природної мови (Natural Language Processing, NLP) та глибокого навчання. Нейронні мережі, такі як рекурентні нейронні мережі або трансформери, можуть використовуватись для аналізу та розуміння текстових вхідних даних;

– після розуміння мови голосовий асистент повинен здатися зрозумілим користувачу. Це можна досягти за допомогою генерації тексту або голосової синтеза. Для генерації тексту можуть використовуватись глибокі нейронні мережі, такі як рекурентні нейронні мережі або трансформери, які генерують відповідний текст на основі розуміння вхідного запиту. Для генерації голосу можуть використовуватись техніки голосового синтезу на основі нейромереж. Це може включати в себе використання глибоких генеративних моделей, таких як глибокі згенеративні моделі автокодування (Deep Generative Autoencoders), глибокі генеративно-показові мережі (Deep Generative AdversarialNetworks) або згенеративні моделі на основі потоків (Flow-based Generative Models), які навчаються створювати голосові сигнали з тексту або фонем;

– персоналізація та навчання з підсиленням: Нейромережі дозволяють голосовим асистентам підлаштовуватись до вподобань та потреб користувача. За допомогою технік навчання з підсиленням (Reinforcement Learning), асистент може навчатись на основі

зворотного зв'язку від користувача і вдосконалювати свої навички з часом. Це дозволяє покращувати якість відповідей та забезпечувати персоналізований досвід користувача;

– інтеграція з іншими системами: Голосові асистенти можуть взаємодіяти з іншими системами та сервісами, щоб надати користувачеві більш широкий функціонал. Використання нейромереж дозволяє забезпечити зручну та ефективну інтеграцію з іншими інформаційними системами, веб-сервісами, базами даних тощо.

Завдяки нейромережам, голосові асистенти стають більш ефективними у розпізнаванні мови, що дозволяє їм точніше розуміти команди користувача. Вони можуть розпізнавати не тільки окремі слова, але й контекст, інтонацію, емоції та інші аспекти мови, що покращує якість і точність розуміння.

Голосові асистенти, розроблені з використанням нейромереж, можуть бути персоналізовані та навчені. Вони можуть враховувати індивідуальні вподобання, контекст та стилі мовлення користувача. За допомогою технік навчання з підсиленням, асистенти можуть покращувати свої навички з часом і надавати більш точні та релевантні відповіді.

1.3 Які проблему вирішує створення голосового асистента

У сучасному світі, де технології стають все більш інтегральною частиною нашого повсякденного життя, голосові асистенти стають все більш популярними і корисними інструментами. Голосові асистенти – це програмні рішення, засновані на штучному інтелекті і машинному навчанні, які взаємодіють з користувачами за допомогою голосових команд і надають широкий спектр послуг та функцій.

Створення голосового асистента вирішує ряд проблем і пропонує рішення, які полегшують наше життя. Ось деякі з них:

– зручність і швидкість: Голосові асистенти дозволяють нам взаємодіяти з технологією за допомогою голосових команд, що значно зручніше, ніж писати або вводити текст. Це особливо корисно в ситуаціях, коли ми зайняті або не маємо можливості використовувати руки (наприклад, під час водіння або при виконанні робіт по дому);

– підвищення продуктивності: Голосові асистенти можуть виконувати різні завдання і допомагати з організацією робочого часу. Вони можуть створювати нагадування, встановлювати будильники, надсилати повідомлення, здійснювати пошук в Інтернеті, запускати додатки та багато іншого. Це дозволяє зосередитися на важливих завданнях і ефективно керувати своїм часом;

– доступність для людей з обмеженими можливостями: Голосові асистенти відкривають нові можливості для людей з обмеженими можливостями, таких як люди з вадами зору або руховими порушеннями. Вони можуть використовувати голосові команди, щоб керувати своїми пристроями, отримувати інформацію та виконувати завдання без необхідності фізичного контакту з пристроями. Це робить технологію більш доступною і сприяє інклюзивності;

– покращення взаємодії з технологіями: Голосові асистенти відкривають нові способи взаємодії з технологіями. Замість традиційного вводу і навігації за допомогою клавіатури або екрану, користувачі можуть просто вимовити команду або запит, і голосовий асистент виконає його. Це створює більш природний та інтуїтивний спосіб взаємодії з технологіями, зменшуючи бар'єри і сприяючи легкості використання;

– розвиток інтелектуального домашнього середовища: Голосові асистенти відіграють важливу роль у розвитку інтелектуального домашнього середовища. Вони можуть керувати підключеними пристроями, такими як освітлення, термостати, безпекові системи тощо.

Користувачі можуть за допомогою голосових команд контролювати свої домашні пристрої, забезпечуючи зручність, енергоефективність та безпеку.

Створення голосових асистентів вирішує ряд проблем і надає користувачам широкий спектр переваг. Вони полегшують нам взаємодію з технологією, підвищують продуктивність, сприяють доступності для людей з обмеженими можливостями, покращують взаємодію з технологіями і сприяють розвитку інтелектуального домашнього середовища. Вони стають невід'ємною частиною нашого повсякденного життя, допомагаючи нам здійснювати різні завдання швидше, зручніше і ефективніше.

Однак, наряду з перевагами, існують також питання, пов'язані з приватністю та безпекою даних, а також залежністю від технології. Важливо розуміти ці аспекти і враховувати їх при використанні голосових асистентів. Тому було вирішено створити персонального голосового асистента якого користувач зможе настроїти під свої необхідності.

Загалом, створення голосових асистентів розв'язує ряд проблем і пропонує користувачам нові способи взаємодії з технологіями. Вони допомагають нам бути більш продуктивними, доступними та зручними, відкривають нові можливості і сприяють розвитку інноваційного технологічного середовища. Зростаюча популярність голосових асистентів свідчить про їх значимість і потенціал у покращенні нашого життя.

1.4 Аналіз існуючих програмних рішень

В цьому розділі будемо розглядати декілька самих відомих примірів голосових асистентів для їх аналізу та порівняння їх функціональності. На разі такими є Google Assistent, Cortana, Silvius та Alexa. Кожен цей застосунок має як свої переваги так і свої мінуси, для початку потрібно ознайомитися з кожним і зробити висновки які мають допогти при створенні власного голосового асистенту.

1.4.1 Голосовий помічник від Google

Голосовий помічник від компанії «Google», відомий як Google Assistant, доступний як для мобільних платформ, так і для розумних домашніх пристроїв. Він також інтегрується з пристроями розумного будинку. Користувачі взаємодіють з Google Assistant переважно за допомогою голосових команд, хоча є можливість використовувати його за допомогою клавіатури [27].

Google Assistant надає голосове керування та можливості пошуку, що дозволяє користувачам виконувати різноманітні завдання: керування розумним домом, доступ до особистої інформації з сервісів Google, пошук інформації в Інтернеті, бронювання ресторанів або квитків, перегляд погоди та новин, прослуховування музики, встановлення таймерів та нагадувань, організація зустрічей, запуск програм на пристроях, обмін повідомленнями, реальний час перекладів тощо. Google Assistant може розуміти контекст, в якому задається питання, що дозволяє йому забезпечити більш точні та природні відповіді. В майбутньому, за заявами Google, помічник зможе самостійно телефонувати та бронювати зустрічі від імені користувачів [27].

1.4.2 Голосовий помічник від Amazon «Alexa»

Alexa, розроблений і підтримуваний компанією «Amazon», є багатофункціональним голосовим асистентом. Він пропонує широкий спектр функцій, таких як відтворення музики і аудіокниг, налаштування будильників, створення списків завдань, надання актуальної інформації про погоду, трафік та новини. Користувач може легко взаємодіяти з асистентом, використовуючи голосові команди.

Alexa також широко застосовується у сфері Інтернету речей, де вона дозволяє керувати автоматизованою домашньою системою розумних пристроїв. Користувачі можуть розширити можливості Alexa,

встановлюючи наявні навички або програмуючи власні «навички» – додаткові функціональні можливості, які створюються зовнішніми розробниками. Створені користувачами «навички» можуть бути опубліковані в системі Alexa Skills, інші користувачі зможуть користуватись ними на своїх пристроях.

Багато «навичок» запускаються повністю в хмарному середовищі, використовуючи всі можливості веб-сервісів Amazon.

1.4.3 Голосовий помічник «Silvius»

Silvius є нащадком Dragon NaturallySpeaking і Aenea і використовує власний фреймворк, відомий як Kaldi, для розпізнавання мовлення. Цей голосовий асистент може працювати як у режимі онлайн, так і на вбудованих пристроях. Система Silvius передає аудіосигнал від мікрофона на сервер, який відповідає зрозумілим текстом, отриманим з розпізнавання. Отриманий текст проходить через граматику та імітує натискання відповідних клавіш на клавіатурі. Незважаючи на використання мережевого зв'язку з сервером, голосовий асистент демонструє високу швидкість відгуку. Головною інновацією Silvius є його здатність до розпізнавання мовлення без залежності від конкретної платформи, що дозволяє використовувати голосового асистента на будь-якій операційній системі. Іншою перевагою системи є її безкоштовний характер та відкритість тексту програми для розширення та модифікації. Встановлення системи також відносно просте. При розробці використовувалися мова програмування Python та фреймворк для аналізу даних Spark. В голосовому асистенті Silvius, так само як у VoiceCode, користувачі вводять команди не у звичній розмовній формі, а за допомогою спеціальних слів, що вимагає певного часу для оволодіння ними. Команди, надані системою, зазвичай не є високорівневими і зводяться до простого диктування програмного коду

розробником. Однак, одним з недоліків є недостатня документація, що ускладнює роботу користувачів з цією системою та збільшує час, необхідний для початку роботи з нею.

1.4.4 Голосовий помічник «Cortana»

Функції особистого помічника Кортани включають передбачення потреб користувача. Користувач має можливість надати доступ до своїх особистих даних, таких як електронна пошта, адресна книга, історія пошуку в Інтернеті та інші, і Кортана використовує ці дані для задоволення потреб користувача. Кортана може замінити стандартну пошукову систему та активується натисканням кнопки «Пошук». Користувач може вводити запити вручну або голосом. Вона здійснює пошук інформації в системах Bing, Foursquare та в особистих файлах користувача. Крім того, вона має почуття гумору і може проводити бесіди, співати та розповідати анекдоти. Кортана також нагадує про заплановані зустрічі, дні народження інших людей та інші важливі події. Інтерфейс має гнучкі налаштування приватності, що дозволяють користувачеві самостійно визначати, які дані надавати віртуальному помічнику. Розробники стверджують, що рівень контролю над приватністю, який пропонує Кортана, перевершує Siri та Google Now [30].

Кортана використовує браузер Edge та пошукову систему Bing, і не підтримує інші браузери. Вона також інтегрується з деякими програмами з Windows Store.Кортана використовує браузер Edge та пошукову систему Bing та не підтримує інші браузери. Вона інтегрується з деякими програмами з Windows Store [30].

2.4.5 Висновки

Результати проведеного аналізу вказують на те що кожна з цих програм мають недоліки та їх рішення не задовільняє більшості з висунутих вимог.

Як видно з дослідження Alexa, Google Assistant та Cortrana – це голосові помічники загального призначення, але вони не орієнтований на вирішення проблеми написання тексту програми за допомогою голосу. Це не просто системи, а платформи, за допомогою яких інші розробники можуть інтегрувати та використовувати їх у своїх цілях. Кожна з цих систем добре документована та підтримується. найбільшими IT-компаній. Однак іншим недоліком є те, що вони не безкоштовні та вимагають грошових вкладень для їх використання. Також вони мають проблеми з конфедецйністтю данних, власники цих застосунків збирають інформацію користувачів (таблиця 1.1).

Таблиця 1.1 – Оцінювання аналогів по критеріям

Критерії оцінювання застосунку	Google Assistant	Cortana	Silvius	Alexa
Можливість на перетворення тексту програми голосом	-	-	+	-
Надає відкритий текст програми та є безтоштовним	-	-	+	-
Модульність та кросплатформеність	+	+	-	+
Наявна документація для користування	+	+	-	+
Конфенедційність	-	-	+	-

По проведеному аналізу та висновкам зрозуміло, що ці застосунки мають проблеми. Тому необхідно створити програмний засіб для вирішення всіх цих та не вказаних задач для управління інтегрованими середовищами розроблення програмного забезпечення, яке не міститиме недоліки розглянутих рішень та використає їх сильні сторони.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета дослідження

Метою дослідження є створення застосунку для персональних комп’ютерів та телефоні на основі нейронних мереж, який зможе допомагати користувачам щвидше та простіше користуватися своїм пристроєм та виконувати дії через голосові команди. Також, дослідження проводится з ціллю виявити можливості використання нейронних мереж для створення штучного інтелекту для голосового помічника. В цей процес входить ствоерння експериментального прототипу використовуючи штучні алгоритми та нейронні мережі. Основним завданням дослідження є покращення точності існуючих методів та підходів до ідентифікації іменованих сутностей тексту програми.

2.2 Завдання дослідження

Щоб виявити основні проблеми та способи її вирішення для досягнення мети дослідження потрібно виконати декілька задач:

– провести аналіз відомих застосунків та визначити основні проблеми, які були не вирішенні і знайти єфективні методи для їх рішення;

– провести аналіз існуючих нейронних мереж та визначити їх сильні і слабі сторони;

– проведення тестів неронних мереж для визначення необхідних;

– обрати необхідне середовище для створення прототипу Голосового помічника;

– визначення основних задач та функціоналу необідних для різних типів користувачів;

– розробка програмного прототипу та проведення

єксперементального дослідження для перевірки функціональності та єфективності застосунку.

2.3 Практична цінність

Створений голосовий асистент на базі нейронних мереж відкриває для користувачів наступні можливості:

– простота у використанні. Кожен користувач зможе використовувати голосового помічника на будь-якому пристрої що піддержує необхідні технології;

– підвищує продуктивність роботи, оскільки голос є найбільш ефективним засобом комунікації;

– конфеденційність даних та можливість використовувати помічника без доступу до інтернету;

– дозволяє людям з обмеженими можливостями мати доступ для необхідної інформації та керування програмами;

– автоматизація рутинних завдань для економії часу;

– розпізнавання користувачів через технології нейронних мереж.

3 КОМП'ЮТЕРНА СИСТЕМА

3.1 Технічні характеристики комп'ютера та зовнішніх пристроїв

Розробка даного програмного забезпечення проводилась на комп'ютері з даними характеристиками: ноутбук, об'єм оперативної пам'яті – 16 Гбайт, швидкість процесору – 2.6 ГГц, 6 ядер та 12 потоків, монітор з розширенням 2880x1800.

Для роботи розробленого програмного забезпечення повинніви користовуватися наступні технічні засоби: Персональний комп'ютер (системний блок, монітор, клавіатура, мишка, мікрофон).

Характеристики:

- процессор 1.4 ГГц;
- оперативна пам'ять 512 Мбайт;
- кількість доступної пам'яті на накопичувачі більше 500МБ;
- операційна система на ядрі Windows;
- звукова карта;
- мікрофон.

Рекомендовані вимоги до комп'ютера:

- процессор 1.5 ГГц і більше;
- оперативна пам'ять – 2056 Мбайт і більше;
- кількість доступної пам'яті на накопичувачі більше 1ГБ;
- операційна система на ядрі Windows;
- звукова карта;
- мікрофон.

3.2 Вибір програмних засобів та операційної системи

Усі елементи керування розробляються власноруч. Передача даних між клієнтом та сервісом є важким організаційним процесом. Але є велика кількість значущих переваг такого методу:

− легкість розбиття на модулі та програмування кожного з них окремо один він одного;

− з особливостей вище витікає легкість процесу відладки;

− дуже велика гнучкість та легкість внесення змін як простих, так і об’ємних;

− незалежність виконання від встановленої операційної системи;

− легкість проектування програми під особисті потреби користувача.

3.2.1 Мова програмування

Для реалізації задачі було обрано мову програмування Python. Ідея Python виникла в 1989 році, коли її творець Гвідо ван Россум зіткнувся з недоліками мови ABC (а саме з розширюваністю). Россум розпочав роботу над розробкою нової мови, яка інтегрувала всі корисні функції мови ABC та нові бажані функції, такі як розширюваність та обробка виключень. Python 1.0 вийшов у 1994 році; вона запозичила модульну систему від Modula-3, мала можливість взаємодіяти з операційною системою Amoeba і включала функціональні засоби програмування [1].

У 2000 році основна команда розробників Python перейшла на Beopen.com, а в жовтні 2000 року Python 2.0 був випущений з багатьма імпровізаціями, включаючи сміттєзбірник та підтримку Unicode.

У грудні 2008 року було випущено Python 3.0, відмовившись від зворотної сумісності та нового дизайну, щоб уникнути дублювання конструкцій та модулів. Це все ще багатопарадигмальна мова, яка пропонує розробникам параметри об'єктноорієнтованого, структурованого програмування та функціонального програмування [3].

Сьогодні Python має декілька реалізацій, включаючи Jython, написаний мовою Java для віртуальної машини Java; IronPython написаний на C# для загальної мовної інфраструктури, а версія PyPy написана в

RPython та переведена на C. Хоча ці реалізації працюють на рідній мові, на якій вони написані, вони також можуть взаємодіяти з іншими мовами за допомогою модулів. Більшість із цих модулів працюють за моделлю розвитку громади та є відкритими та безкоштовними [3].

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках. Є реалізація інтерпретатора для JVM з можливістю компіляції, CLR, LLVM, інші незалежні реалізації. Проект PyPy використовує JIT-компіляцію, яка значно збільшує швидкість виконання Python-програм [14].

Компанії, що розробляють програмне забезпечення, віддають перевагу мові Python через його універсальні можливості та меншу кількість програмних кодів. Майже 14% програмістів використовують його в таких операційних системах, як UNIX, Linux, Windows та Mac OS. Програмісти великих компаній використовують Python, оскільки він створив собі позначку в розробці програмного забезпечення з характерними функціями, такими як:

- інтерактивність;
- інтерпретованість;
- модулярність;
- динамічність;
- об’єктно-орієнтований стиль;
- портатиність;
- високорівневість;
- розширення з C та C++.

Також, ця мова надає великі стандартні бібліотеки. Більшість широко використовуваних завдань програмування вже написані в цих бібліотеках, що збільшує компактність коду.

Більш того, поріг входження низький, а код багато в чому лаконічний і зрозумілий навіть тому, хто ніколи його не використовував. За рахунок простоти коду подальший супровід програм, написаних на Python, стає легше і приємніше в порівнянні з Java або C++. З точки зору бізнесу перевага полягає у скороченні витрат і збільшенні продуктивності праці співробітників.

Крім сотні тисяч індивідуальних розробників і невеликих компаній, Python підтримують такі гіганти IT як:

– Facebook;
– Yandex;
– RedHat;
– Google;
– Dropbox;
– Telegram;
– Mozilla;
– Amazon.

І саме головне, компанія Google надає дуже зручні API клієнти для Python.

Підвівши підсумки, маємо, що Python – це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читання коду; синтаксис ядра Python мінімалістичний; це мова програмування, що спеціально розроблена для написання швидкодіючих програм; є підтримка структурного, об'єктно - орієнтованого, функціонального, імперативного і аспектноорієнтованого програмування; основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопотокових обчислень, високорівневі структури даних; підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

3.2.2 Інтегроване середовище розробки

Інтегроване середовище розробки – це програмне забезпечення, що забезпечує користувальницький інтерфейс для розробки коду, тестування та налагодження функцій. Інтегроване середовище розробки надає декілька інструментів та функцій для полегшення розробки та стандартизації на основі мови програмування, на якій розробник пише програмний код. Також є функції компіляції та інтерпретації програми. Деякі з широко використовуваних ідентифікаторів – це Eclipse для розробки програм програмування Java, Microsoft Visual Studio, Android Studio для розробки мобільних додатків, RStudio для програм R та PyCharm для програмування Python [16].

За допомогою інтегрованого середовища розробки можна розробляти додатки, або динамічні веб-програми тощо. Воно включає редактор коду, компілятор або інтерпретатор та відладчик, щоб отримати доступ до графічного інтерфейсу користувача та дозволяє користувачеві писати та редагувати вихідний код у редактор коду.

Основна мета використання полягає в тому, щоб воно дозволило швидко та ефективно писати програмний код. Як вже було сказано вище, середовище включає вбудовані компілятори, які перетворюють програму в код машинного рівня або байткод і економлять багато часу. Є декілька мов програмування на вибір але я обрав PyCharm.

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python. Ось деякі переваги використання PyCharm:

– потужний редактор коду: PyCharm має функціональний та інтуїтивно зрозумілий редактор коду з підсвічуванням синтаксису, автодоповненням, вирівнюванням коду та багатьма іншими корисними функціями. Він допомагає зробити розробку швидшою та ефективнішою;

– повна підтримка Python: PyCharm надає повну підтримку для всіх функцій та можливостей мови програмування Python. Він розуміє синтаксис, структуру коду та пропонує відповідні підказки, що спрощує процес розробки;

– універсальність: PyCharm підтримує розробку не тільки на Python, але й на інших мовах програмування, таких як JavaScript, HTML, CSS і багатьох інших. Це дозволяє зручно працювати з багатомовними проектами та забезпечує більшу гнучкість в роботі;

– усунення помилок: PyCharm надає потужні інструменти для виявлення та усунення помилок у коді. Він автоматично перевіряє синтаксис, виконує статичний аналіз, надає підказки та підсвічує можливі проблеми, що допомагає зробити код більш надійним та стабільним.

4 ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ

4.1 Огляд та вибір способу розпізнавання голосу в програмі

У пайтоні є кілька бібліотек, які можна використовувати для розпізнавання голосу. Ось декілька з них:

– SpeechRecognition: Ця бібліотека дозволяє розпізнавати голосові команди за допомогою різних бекендів, таких як Google Speech Recognition, CMU Sphinx та інших. Вона надає простий спосіб отримати текстову інформацію з аудіофайлу або мікрофону;

– PocketSphinx: Це Python-інтерфейс для CMU Sphinx, який є одним з популярних відкритих інструментів розпізнавання голосу. Він може працювати автономно (offline) і не потребує з'єднання з Інтернетом;

– DeepSpeech: Це бібліотека розпізнавання мови на основі глибокого навчання. Вона використовує нейромережі для розпізнавання мови з аудіофайлів або мікрофону. DeepSpeech була розроблена Mozilla і має готові моделі, які можна використовувати;

– Kaldi: Це бібліотека розпізнавання мови з відкритим кодом, яка надає широкі можливості для розпізнавання голосу. Вона може бути трохи складнішою у використанні, але має велику функціональність і підтримку.

Мною була обрана бібліотека SpeechRecognition по деяким причинам які підходять для мого проекту:

– простота використання: SpeechRecognition надає простий та зрозумілий інтерфейс для розпізнавання голосу. Можно легко і швидко інтегрувати її в свій проект і почати працювати з розпізнаванням голосу всього за кілька рядків коду;

– підтримка різних бекендів: SpeechRecognition дозволяє використовувати різні бекенди для розпізнавання, такі як Google Speech

Recognition, CMU Sphinx, Wit.ai та інші. Це дає можливість вибрати найбільш підходящий для ваших потреб сервіс розпізнавання;

– інтернет-підтримка: Якщо потрібно використовувати розпізнавання голосу з Інтернету, наприклад, для інтерактивних додатків або веб-сервісів, SpeechRecognition може бути відмінним варіантом. Можно легко підключитися до сервісів розпізнавання голосу через API та отримувати текстові результати;

– підтримка мов: SpeechRecognition підтримує розпізнавання голосу на різних мовах, що робить його універсальним інструментом для розпізнавання голосу в різних культурних контекстах.

Загалом, SpeechRecognition є зручним і легким у використанні інструментом, який забезпечує широкі можливості для розпізнавання голосу у вашому проекті.

Ось так виглядає запит до програми за допомогою SpeechRecognition (рисунок 4.1).

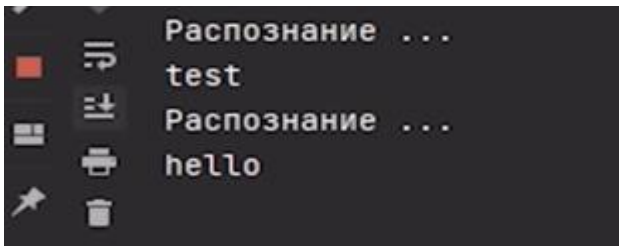

Рисунок 4.1 Розпізнавання мовлення

Наразі програма вміє тільки переводити аудіо в текс та розпізнає мову лише англійською тому потрібно буде використати модель для того щоб програма мала змогу розмовляти.

Весь процес обробки мовного сигналу можна розбити на кілька основних етапів:

– передобробка сигналу;
– виділення критеріїв;
– розпізнавання диктора.

Основні риси голосу формуються трьома основними якостями: механікою коливань голосових складок, анатомією мовного тракту і системою управління артикуляцією. Головні ознаки, якими приймається рішення про особистість диктора, формуються з урахуванням всіх чинників процесу мовлення: голосового джерела, резонансних частот мовного тракту та його згасань, і навіть динамікою управління артикуляцією. Якщо розглянути джерела докладніше, то властивості голосового джерела входять: середня частота основного тону, контур і флюктуації частоти основного тону і форма імпульсу збудження. Спектральні характеристики мовного тракту описуються огинаючим спектром та його середнім нахилом, формантними частотами, довготривалим спектром або кепстром. Крім того, розглядається також тривалість слів, ритм (розподіл наголосів), рівень сигналу, частота та тривалість пауз. Щоб визначити ці показники доводиться використовувати досить складні алгоритми, але оскільки, наприклад, похибка формантних частот досить велика, спрощення використовуються коефіцієнти кепстра, обчислювані по огинаючої спектра чи передатна функція мовного тракту, знайдена шляхом лінійного прогнозування. Крім згаданих коефіцієнтів кепстру також використовуються їх перші та другі різниці за часом. Цей метод було вперше запропоновано роботах Девіса і Мермельштейна [23].

У роботах з розпізнавання голосу найпопулярніший метод кепстрального перетворення спектра мовних сигналів який використовується і в SpeechRecognition. Схема методу така: на інтервалі часу в 10-20 мс обчислюється поточний спектр потужності, а потім застосовується зворотне перетворення Фур'є від логарифму цього спектра (кепстр) і знаходяться коефіцієнти:

$$c_n = \frac{1}{\theta} \int_0^{\theta} |S\ (j, \omega, t)|^2\ exp^{-jn\omega \mathrm{C}}\ d\omega, \mathrm{C} = 2\frac{2\pi}{\theta}, \theta. \qquad (4.1)$$

верхня частота в спектрі мовного сигналу $|S\ (j, \omega, t)|^2$ – спектр потужності. Число кепстральних коефіцієнтів n залежить від необхідного згладжування спектру і знаходиться в межах від 20 до 40. Якщо використовується гребінка смугових фільтрів, то коефіцієнти дискретного кепстрального перетворення обчислюються як:

$$c_n = \sum_{m=1}^{N} \log Y\ (m)^2 \cos\frac{\pi n}{M}\ (m - \frac{1}{2})), \qquad (4.2)$$

де Y(m) – вихідний сигнал m-го фільтра;

c_n – n-й коефіцієнт кепстру.

Властивості слуху враховуються шляхом нелінійного перетворення шкали частот, зазвичай, у шкалі крейд. Ця шкала формується виходячи з присутності в слуху так званих критичних смуг, таких, що сигнали будь-якої частоти в межах критичної смуги невиразні. Шкала крейда обчислюється як:

$$M\ (f) = 1125 \ln(1 + \frac{f}{700}), \qquad (4.3)$$

де f – Частота в Гц;

M – Частота в крейдах.

Або використовується інша шкала – барк, така, що різниця між двома частотами, що дорівнює критичній смузі, дорівнює 1 барк. Частота B обчислюється як:

$$B = 13\, arctg(0,00076f) + 3{,}5\, artcg\frac{f}{7500}. \quad (4.4)$$

Знайдені коефіцієнти в літературі іноді позначаються як MFCC Mel Frequiency Cepstral Coefficients. Їх число лежить у діапазоні від 10 до 30. Використання перших та других різниць за часом кепстральних коефіцієнтів втричі збільшує розмірність простору прийняття рішень, але покращує ефективність розпізнавання диктора [23].

Кепстр визначає форму огинаючої спектра сигналу, яку впливають і властивості джерела збудження, і особливості мовного тракту. В експериментах було встановлено, що спектр, що огинає, сильно впливає на впізнаваність голосу. Тому використання різних способів аналізу огинаючої спектра з метою розпізнавання голосу цілком виправдано.

4.2 Налаштування синтезу мови

Для початку потрібно розібратися як працює синтез мовлення. Роботу систем синтезу мови за текстом зазвичай можна розділити на два етапи, за які відповідають два окремі компоненти – обробка тексту та синтез мови [41].

На першому етапі, зазвичай, проводиться кілька операцій обробки природної мови. Спочатку текст розбивається на окремі речення, а потім речення поділяються на слова. Після цього застосовується нормалізація тексту, що дозволяє привести його до стандартного вигляду. Також проводиться автоматична морфологічна розмітка, що включає в себе визначення граматичних характеристик слів, таких як число, рід, час, особа тощо. Крім того, може застосовуватися конвертація графем у фонеми (G2P), що дозволяє визначити фонетичну структуру слів.

Наступний етап – це етап синтезу мови. На цьому етапі використовується послідовність фонем, отриманих на попередньому етапі, для створення синтезованої звукової хвилі промови. При цьому враховуються алгоритми передбачення просодії, які визначають такі особливості мовлення, як тон, наголос, гучність та інші просодичні ознаки, що доповнюють основну артикуляцію звуку.

В результаті цих етапів отримується синтезований голосовий вихід, що максимально відтворює природну мову з усіма її лінгвістичними особливостями.

Зазвичай синтезатори мови не працюють безпосередньо з сигналом звукової хвилі, а використовують деяке уявлення цього сигналу, наприклад, спектрограму. Алгоритм, здатний виділити такі параметри і по них відновити звукову хвилю, називається вокодер. Прикладами таких алгоритмів є мю-закон та відновлення сигналу за його спектрограмою.

Мю-закон перетворює кожне значення дискретизованого сигналу:

$$x = \{x_1, x_2, \dots, x_t\} \text{ як } f(x_t) = sign(x_t)\ \frac{\ln(1+\mu|x_t|)}{\ln(1+\mu)}, \qquad (4.5)$$

де $\mu = 255$ та $-1 < x_t < 1$, а потім кодує сигнал як двозначне шістнадцяткове число, яке позначає деякий інтервал на числовій прямій. Зворотне перетворення вибирає значення перетвореного сигналу $y_t = f(x_t)$ з даного номера інтервалу та отримує оцінку вихідного сигналу x наступним чином:

$$f^{-1}(y_t) = sign(y_t)(1/\mu)(\left(1+\mu\right)^{|y_t|} - 1), -1 < y_t < 1. \qquad (4.6)$$

Як уявлення звукового сигналу може бути спектрограма хвилі – зображення, що показує залежність спектральної щільності потужності

сигналу від часу. Спектрограми відображають частоту та амплітуду сигналу в часі, але не містять жодної інформації про фазу сигналу, через що результат зворотного відновлення сигналу спектрограмі неминуче відрізняється від оригіналу. Цифрова генерація спектрограми сигналу s(t) зводиться до обчислення квадрата амплітуди віконного перетворення Фур'є.

$$spectogram(t, \omega) = |STFT(t, \omega)|^2, \quad (4.7)$$

де ω використовуване вікно;

$STFT(t, \omega)$ – віконне перетворення Фур'є.

Давайте розглянемо дві популярні мережі, які використовуються для перетворення тексту на спектрограму і синтезу промови за допомогою спектрограми – Tacotron і WaveNet. Ці моделі представляють собою потужні інструменти у галузі синтезу мови.

WaveNet, з іншого боку, є авторегресійною моделлю, що прямує від спектрограми до звукової хвилі промови. Вона працює безпосередньо з мел-спектрограмою і може генерувати дуже реалістичні звуки промови.

Tacotron є авторегресійною моделлю, яка перетворює текстовий ввід на спектрограму. Вона працює в двох етапах: спочатку перетворює текст на мел-спектрограму (представлення амплітуди звуку на різних частотах), а потім генерує спектрограму для синтезу промови.

Tacotron – модель параметричного синтезу мови, заснована на підході seq2seq, розроблена Google та опублікована у 2017 році. Модель складається з кодера, декодера з увагою та нейронної мережі для пост-процесингу сигналу. Схему моделі зображено на рисунку 4.2.

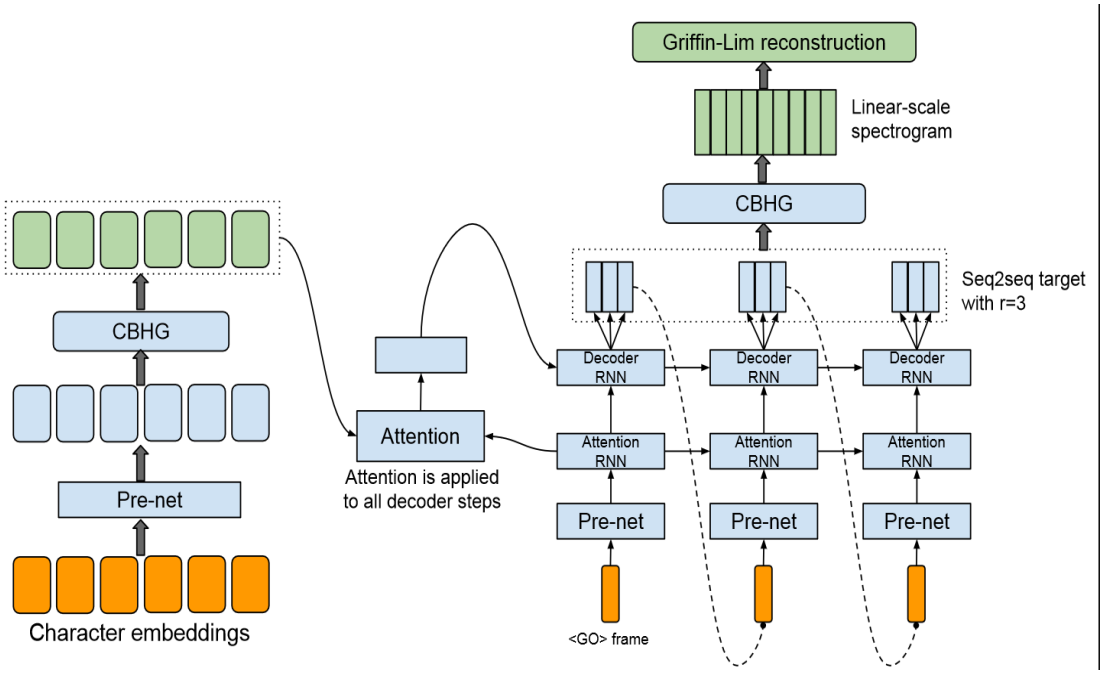

Рисунок 4.2 – будова моделі Tacotron

Кодер і мережу пост-процесингу спираються на блок CBHG, схема якого зображена на Малюнку 2. Блок складається з набору одновимірних згорткових фільтрів, за якими слідують шосейні нейронні мережі, що є модифікацією LSTM мереж, і двонаправлений керований рекурентний блок. Вхідна послідовність спочатку обробляється K наборами згорткових фільтрів з розмірністю 1,2,…,K. Ці фільтри моделюють локальну та контекстно-залежну інформацію (за аналогією з моделюванням уніграм, біграм, аж до K -Г). Вихід згорткового рівня далі обробляється шосейною мережею подальшого виділення параметрів. Наприкінці CBHG використовується керований рекурентний блок, який для виділення параметрів спирається на контекст перед аналізованим символом і після символу.

Кодер виконує завдання видалення послідовних параметрів з тексту. Він отримує послідовність символів, які кодуються за допомогою one-hot кодування і об'єднуються у безперервний вектор. Цей вхід проходить через

нелінійні перетворення, які виконуються у мережі з пляшковим шийкою (bottleneck layer), що поліпшує узагальнення мережі та прискорює збіжність. Модуль CBHG перетворює вихід нелінійних перетворень на підсумкове представлення тексту, яке передається в декодер.

Декодер є рекурентною нейронною мережею з увагою, де запит уваги генерується на кожному кроці часу. Вектор контексту об'єднується з виходом уваги і подається на вхід декодуючим рекурентним нейронним мережам, які використовують керовані рекурентні блоки. Вихід декодера представляє собою спектрограму звукової хвилі, яка подається на вхід модулю пост-обробки для генерації безпосередньо звукової хвилі.

Для пост-обробки використовується модуль CBHG, який був описаний раніше. Після цього спектрограма звукової хвилі передається алгоритму Гріффіна-Ліма, який генерує остаточну звукову хвилю.

WaveNet є моделлю, що породжує, використовує параметричний підхід до синтезу мови. Її завдання – відновити розподіл ймовірностей звукового сигналу $x = \{x_1, x_2, \dots, x_t\}$ за допомогою добутку умовних ймовірностей:

$$p(x) = \prod_{t=1}^{T} p\left(x_t \middle| x = \{x_1, x_2, \dots, x_{t-1}\}\right). \quad (4.8)$$

Таким чином, ймовірність кожного сигналу x_t залежить тільки від попередніх сигналів. Під час навчання модель оцінює цей умовний розподіл ймовірностей, приймаючи на вхід сигнали з навчальної вибірки одночасно. На етапі генерації модель породжує вихідні сигнали за допомогою отриманої оцінки розподілу ймовірності послідовно – отриманий сигнал моделлю в момент часу t подається назад на вхід для генерації наступного сигналу в момент часу t+1. Структура моделі дозволяє використовувати її у широкому спектрі завдань, наприклад, у завданнях продовження музичного

твору на його початку, породження голосу конкретної людини або text-to-speech синтезу [41].

Модель WaveNet складається з безлічі згорткових нейронних мереж, аналогічно до моделі PixelCNN. В якості вокодера використовується мю-закон. На рисунку 4.3 зображена повна структура моделі. На вході модель отримує послідовність сигналів, закодованих мю-законом, позначених як x. Опціонально, модель може приймати додаткову інформацію, позначену як вектор параметрів h. На виході модель повертає розподіл ймовірностей параметрів мю-закону, за якими можна відновити синтезований сигнал.

Під час навчання, вхідний сигнал x є прикладом звуку з навчальної вибірки, і він подається на всі входи моделі одночасно. Під час генерації, вхідними сигналами моделі x є сигнали, які вона згенерувала на попередніх кроках часу, і вони передаються їй послідовно.

Додаткова інформація h може містити, наприклад, інформацію про аналізований текст у задачі синтезу мовлення (text-to-speech). Вона допомагає моделі враховувати додаткові контекстуальні відомості під час синтезу звуку.

Таким чином, модель WaveNet використовує мю-закон як вокодер, виконує безліч згорткових операцій і приймає закодовані сигнали на вході, щоб згенерувати розподіл ймовірностей параметрів мю-закону для синтезу сигналу.

Модель є набір з K блоків з залишковим зв'язком, що містять перетворення розширеної причинної згортки та функцію активації. Функція активації у своїй аналогічна функції, запропонованої моделі Gated PixelCNN.

$$y = tahn\left(W_{f,k} * x\right) \odot \sigma\left(W_{g,k} * x\right), \tag{4.9}$$

де k – номер шару моделі;

f, g – індекси входів перетворення;

σ – функція сигмоїди;

∗ – операція згортки;

⊙ – операція покомпонентного множення векторів;

W – вивчені параметри згортки.

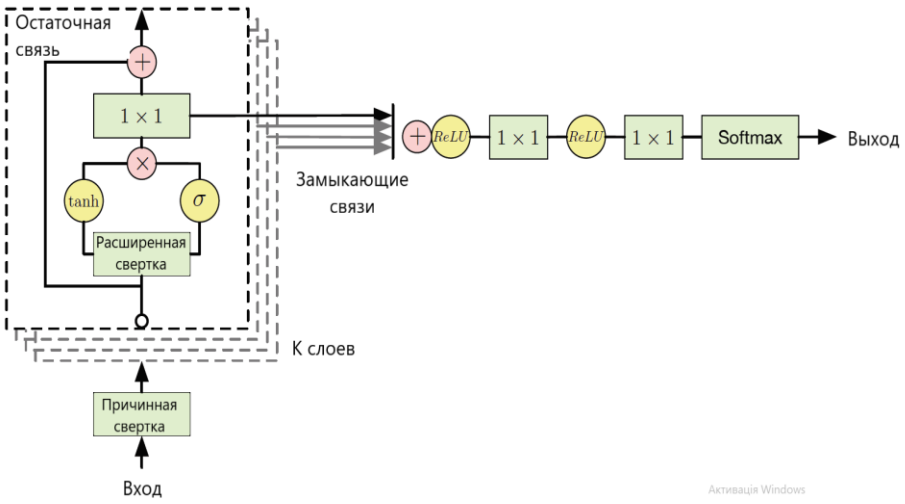

Рисунок 4.3 будова моделі WaveNet

У разі використання додаткової інформації h модель оцінює розподіл ймовірностей звукового сигналу як:

$$p(x|h) = \prod_{t=1}^{T} p\left(x_t|\ x = \{x_1, x_2, \dots, x_{t-1}, h\}\right). \quad (4.10)$$

Для використання додаткової інформації h функція активації може використовувати вектор h глобально при обчисленні кожного згортки як:

$$y = tahn\left(W_{f,k} * x + V_{f,k}^{T} h\right) \odot \sigma\left(W_{g,k} * x + V_{g,k}^{T} h\right), \quad (4.11)$$

де $V_{f,k}$ – вивчені параметри лінійної проекції.

У разі коли розмірність вектора h менше, ніж розмірність x, можна використовувати upsampling-перетворення згорткових мереж, щоб отримати вектор розмірності x і використовувати його замість оригінального вектора h. При цьому $V_{f,k} * h$ буде операцією згортки розміру 1.

Найвідомішим прикладом застосування WaveNet для синтезу мови є технологія Google Асистент, яка використовує WaveNet для генерації голосів асистентів різними мовами. Модель дозволила значно скоротити кількість записів промови акторів озвучки, необхідних створення голосової моделі.

Для задійснення WaveNet чи Tacotrn мені не вистачило часу, тому в прототипі програми для відтворення та запису аудіо в Python була використана бібліотека sounddevice. Вона надає зручний інтерфейс для роботи з аудіопотоками та пристроями вводу/виводу звуку. Декілька основних рис бібліотеки Sounddevice:

Відтворення аудіо: Sounddevice дозволяє відтворювати аудіофайли або звукові дані з масивів прямо з вашого Python-коду. Ви можете відтворювати звук з файлів різних форматів, таких як WAV, FLAC, MP3 та інших.

Запис аудіо: Ви можете використовувати Sounddevice для запису аудіо з мікрофону або інших аудіопристроїв. Вона дозволяє вам отримувати аудіодані у реальному часі та зберігати їх у файл або обробляти негайно.

Низький рівень затримки: Sounddevice має оптимізовану архітектуру, що дозволяє досягти низького рівня затримки аудіо. Це важливо для багатьох додатків, таких як аудіоінтерфейси, реалізація синтезаторів звуку та інших додатків, де мінімізація затримки є критичною.

Підтримка багатьох платформ: Sounddevice підтримує різні операційні системи, такі як Windows, macOS та Linux. Це робить його універсальним вибором для проектів, які працюють на різних платформах.

Простий інтерфейс: Бібліотека Sounddevice має простий і легкий у використанні інтерфейс. Вона надає можливість налаштовувати параметри аудіо, контролювати гучність та інше.

Розглянемо цю структуру синтезу, виділивши у ній всі значущі компоненти, відповідні завданням, вирішуваним у процесі синтезу промови. На рисунку 4.4 представлена структура фонемного синтезатора мови за текстом, основними компонентами якого є:

– лінгвістичний процесор;
– фонетичний процесор;
– просодичний процесор;
– акустичний процесор.

Як очевидно з цієї структури, синтезатор мови складається з низки процесорів (оброблювачів), основне завдання яких полягає у поступовій обробці вхідного орфографічного тексту. Ця обробка є розміткою тексту з метою його перетворення в послідовність параметрів, що описують мовний сигнал. Результуюча послідовність параметрів має бути «сприйнята» акустичним процесором, завданням якого є безпосереднє відтворення звуку, тобто. власне синтез мови. Особливості функціонування кожного з наведених компонентів синтезатора мови залежать від обраної моделі синтезу. При цьому деякі з цих компонентів можуть зникнути, деякі об'єднатися в один, деякі перетворитися в інший. У даному навчальному посібнику говоритимемо про деяку «усереднену» структуру синтезатора мови та враховуватимемо особливості окремих моделей синтезу мови при розгляді тих компонентів, у яких вирішення відповідних завдань буде доречним.

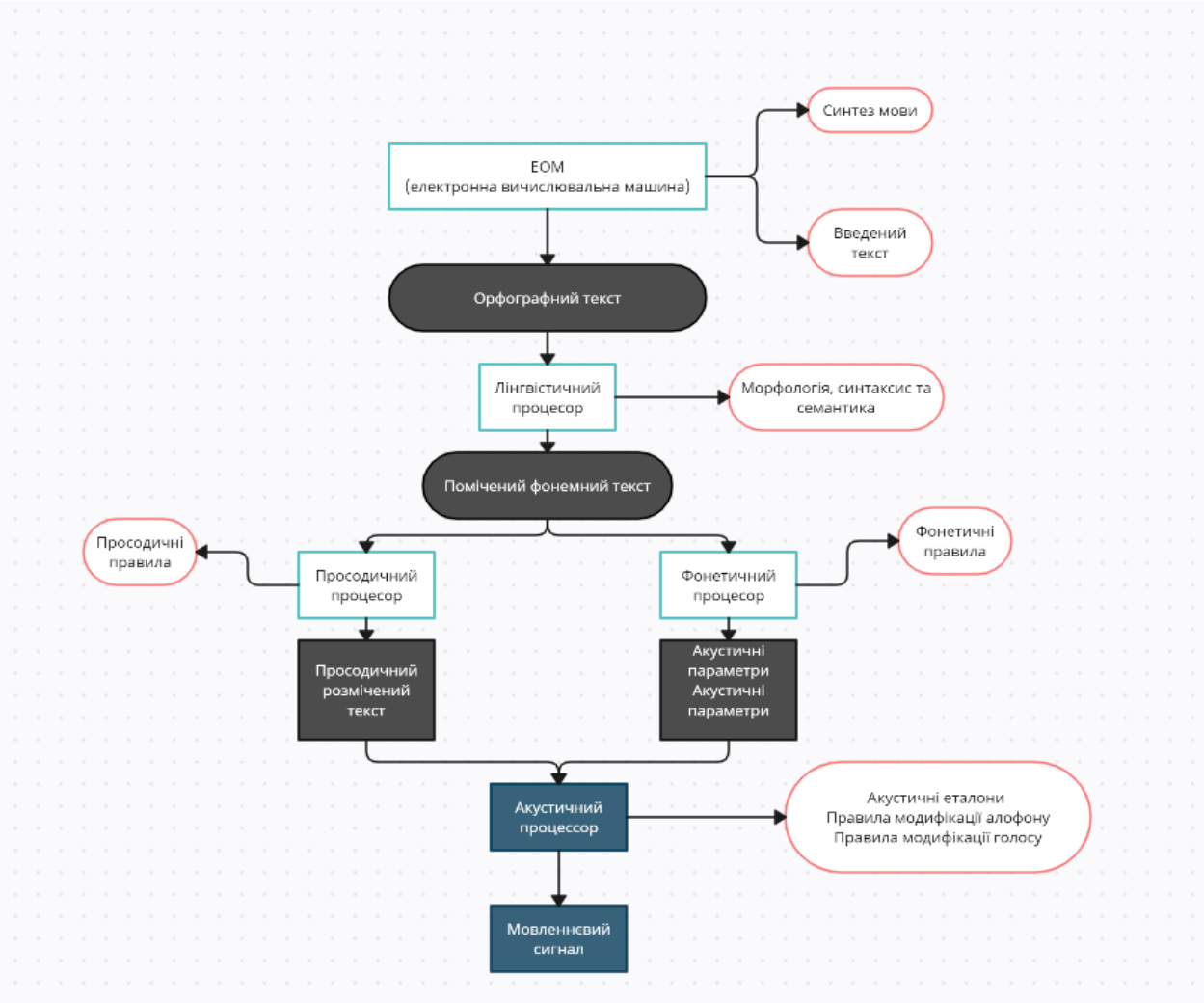

Рисунок 4.4 Структура фонемного синтезатора мови за текстом

Розглянемо коротко подані на рисунку компоненти синтезатора мови. Перший компонент називається лінгвістичним процесором. Іноді його називають текстовим процесором. Лінгвістичний процесор призначений для перетворення вхідного орфографічного тексту на фонемний текст. Зазначений фонемний текст потрапляє в пам'ять ЕОМ одним із двох способів: або його вводить користувач, або він є результатом роботи системи, що автоматично синтезує природно-мовні тексти. Лінгвістичним процесором використовується лінгвістична база даних та знань, що включає словник природної мови, а також знання морфології, синтаксису та семантики. У результаті виході лінгвістичного процесора ми маємо замість

орфографічного тексту розмічений фонемний текст. Під розміткою розуміється розбиття тексту деякі елементи у наступній ієрархії:

– фонетичний період;
– фрази;
– синтагми.

Крім того, лінгвістичний процесор здійснює:

– розміщення наголосів;
– інтонаційне маркування.

Це основні п'ять функцій лінгвістичного процесора. Розмічений фонемний текст надходить на вхід двох наступних процесорів: просодичного та фонетичного. На виході просодичного процесора виходить просодично розмічений фонемний текст. На виході фонетичного процесора виходить або послідовність акустичних параметрів або формантні параметри або алофонний текст. Це від обраного методу реалізації синтезатора промови. Просодичний процесор працює зі спеціальною просодичною базою даних (БД) та правил, фонетичний процесор – з фонетичними правилами.

В результаті роботи просодичного процесора фонемний текст поділяється на акцентні групи (АГ). Далі здійснюється розмітка АГ на елементи акцентних груп (ЕАГ): інтонаційні пред'ядро, ядро та заядро. І нарешті, остання функція просодичного процесора – це встановлення значень інтенсивності або амплітуди (А), тривалості фонем (Т) та частоти основного тону чи мелодики (F0) для кожного ЕАГ. Завдання фонетичного процесора полягає або у генерації послідовності акустичних параметрів мови, або у виробленні формантних параметрів фонем, або у підстановці позиційних та комбінаторних алофонів. Просодично розмічений фонемний текст і параметри, сформовані фонетичним процесором, поєднуються в одному процесорі, який називається акустичним процесором. Акустичний процесор на підставі інформації про те, які алофони потрібно синтезувати,

а також які лінгвістичні характеристики мають бути приписані кожному алофону, синтезує мовний сигнал. Акустичний процесор використовує відповідну БД, в якій зберігаються акустичні еталони алофонів, правила модифікації алофонів і правила модифікації голосу, що синтезується (рисунок 4.5).

Код для синтезу мови та відтворення аудіопотоку за допомогою sounddevice з налаштуванням усіх змінних для оптимальної роботи. Код для відтворення синтезу (рисунок 4.6).

```
language = 'ua'
model_id = 'ua_v3'
sample_rate = 48000
speaker = 'baya'
put_accent = True
put_yoo = True
device = torch.device('cpu')
text = 'Привіт!!!'

model, _ = torch.hub.load(repo_or_dir='snakers4-silero-models-a7e61e1',
                          model='silero_tts',
                          language=language,
                          speaker=model_id)

model.to(device)


audio = model.apply_tts(text=text,
                        speaker=speaker,
                        sample_rate=sample_rate,
                        put_accent=put_accent,
                        put_yo=put_yo)

print(text)

sd.play(audio, sample_rate)
time.sleep(len(audio) / sample_rate)
sd.stop()
```

Рисунок 4.5 – Код синтезу мови

```
Хелпер (v2.0) начал свою работу ...
привіт
як справи
що робиш
чим займався
відкрив браузер
скільки часу
хаубер скільки часу
асистент скільки часу
```

Рисунок 4.6 – Результат роботи коду

4.3 Вибір та тестування засобу для надання команд програмі

Для успішної реалізації голосового асистента необхідно мати надійний механізм розпізнавання та обробки голосу. В цьому розділі розглянуто використання бібліотеки VOSK API для досягнення цієї мети.

VOSK API є відкритою бібліотекою для розпізнавання мови, розробленою спеціально для завдань голосового управління. Вона базується на мовних моделях, побудованих на основі глибоких нейронних мереж, що забезпечують високу точність розпізнавання мови [40].

Переваги використання VOSK API:

– висока точність розпізнавання: VOSK API використовує сучасні алгоритми глибокого навчання, що дозволяють досягти високої точності розпізнавання мови. Це особливо важливо для голосового асистента, оскільки правильне розуміння голосових команд є критичним фактором для коректної відповіді;

– підтримка різних мов: VOSK API підтримує широкий спектр мов, включаючи англійську, іспанську, французьку, німецьку та інші. Це дає можливість створювати голосові асистенти, які розпізнають та реагують на різномовність користувачів;

– легка інтеграція: VOSK API надає зручний інтерфейс програмування додатків (API), який спрощує інтеграцію з вашим голосовим асистентом. Ви можете використовувати VOSK API в різних програмних середовищах, таких як Python, Java, C++, а також на платформах Android та iOS;

– налаштування та розширення: VOSK API надає можливість налаштовувати параметри розпізнавання, такі як швидкість, чутливість до шуму та інші. Це дозволяє підлаштовувати голосового асистента під конкретні умови та вимоги користувача.

VOSK (Voice Operated Speech Kit) використовує нейронні мережі для розпізнавання мови. VOSK API базується на сучасних алгоритмах глибокого навчання, включаючи рекурентні нейронні мережі (RNN) та технології довгої краткочасної пам'яті (LSTM), що забезпечують високу точність розпізнавання голосу.

Глибокі нейронні мережі в VOSK API навчаються на великих обсягах даних, що дозволяє їм виявляти складні шаблони та залежності в звукових сигналах. Вони аналізують аудіо і виконують послідовну обробку сигналу, щоб перетворити його на послідовність текстових символів, які відповідають розпізнаному слову або фразі [40].

Такий підхід дозволяє VOSK API досягти високої точності розпізнавання мови і забезпечує надійну роботу голосового асистента, що використовує цю технологію.

Для прототипу була обрана модель з офіційного сайту AlphaCephei – VOSK на Українській та російській (деякі функції на Українській не піддержуються) мові (рисунок 4.7).

Ukrainian				
vosk-model-small-uk-v3-nano	73M	TBD	Nano model from Speech Recognition for Ukrainian	Apache 2.0
vosk-model-small-uk-v3-small	133M	TBD	Small model from Speech Recognition for Ukrainian	Apache 2.0
vosk-model-uk-v3	343M	TBD	Bigger model from Speech Recognition for Ukrainian	Apache 2.0
vosk-model-uk-v3-lgraph	325M	TBD	Big dynamic model from Speech Recognition for Ukrainian	Apache 2.0

Рисунок 4.7 – Обрана модель для програми

На цьому етапі прототип програми вміє розмовляти, синтезувати мовлення, але ще потрібно реалізувати команди. В подальшому команди за якими буде працювати програма потрібно буде винести в окрему директорію та імпортувати у вигляді модулей (рисунок 4.8).

```
VA_NAME = 'Помощник'

VA_VER = «2.0»

VA_ALIAS = ('помощник', 'асистент', 'помоги', 'друг')

VA_TBR = ('скажи', 'покажи', 'ответь', 'произнеси', 'расскажи', 'сколько')

VA_CMD_LIST = {
    «help»: ('список команд', 'команды', 'что ты умеешь', 'твои навыки',
'навыки'),
    «ctime»: ('время', 'текущее время', 'сейчас времени', 'который час'),
    «joke»: ('расскажи анекдот', 'рассмеши', 'шутка', 'расскажи шутку',
'пошути', 'развесели'),
    «open_browser»: ('открой браузер', 'запусти браузер', 'открой гугл
хром', 'гугл хром'),
    «weather»: ('погода', 'как там на улице', 'сколько градусов за окном'),
    «searh_web»: ('найди', 'посмотри в интернете', 'найди в интернете',
'напиши в интернете'),
    «send_message»: ('напиши смс', 'отправь сообщение', 'напиши сообщение')
}
```

Рисунок 4.8 – Модель для програми

Конфігурація функцій які вміє робити прототип: назвати свої команди, розповісти жарт, відкрити браузер, назвати час, подивитись

погоду, найти щось в інтернеті, відправити смс. В подальшому планується добавлення інших функцій також і для телефонів.

4.4 Користувацький інтерфейс

Для побудови користувацького інтерфейсу було використано бібліотеку PyQt5 для Python, що є оболочкою для фреймворку Qt написанного на C++. Qt фреймворк вперше став загальнодоступним у травні 1995 року. Спочатку його розробили Хаавард Норд (генеральний директор Trolltech) та Ейрік Чамбе-Енг (президент Trolltech) [2].

Таблиця 2.1 – Загальне порівняння графічних інтерфейсів

Фреймворк	Qt Designer	Tkinter	Kivy
Мобільна розробка	Не підтримує мобільну розробку	Не підтримує мобільну розробку	Підтримує розробку для Android та iOS
3D графіка	Має підтримку 3D графіки	Можливо використовувати 3D графіку, але з обмеженими можливостями	Має вбудовану підтримку 3D графіки
Інтеграція з мовами програмування	Підтримує інтеграцію з C++, Python та іншими мовами	Підтримує інтеграцію з Python	Підтримує інтеграцію з Python та іншими мовами

Продовження таблиці 2.1

Швидкість виконання	Дуже швидкий та ефективний	Швидкість виконання може бути помірною, особливо при обробці великої кількості даних	Може бути повільнішим у порівнянні з іншими фреймворками, особливо при складних обчисленнях
Ліцензія	Вільна та комерційна ліцензії доступні	Вільна ліцензія	Вільна ліцензія
GUI	Є широкий набір готових інструментів та віджетів	Базові віджети з можливістю налаштувань	Гнучка система віджетів та можливість створення власних
Крос-платформеність	(Windows, macOS, Linux)	(Windows, macOS, Linux)	(Windows, macOS, Linux, Android, iOS)
Мова	C++	Python	Python

Продовження таблиці 2.1

Графічний дизайнер	Qt Designer має графічний інтерфейс для візуального створення і налаштування інтерфейсу користувача	Tkinter не має графічного дизайнера, але використовує код для створення і налаштування інтерфейсу користувача	Kivy має графічний інтерфейс для візуального створення і налаштування інтерфейсу користувача

Qt 3.0 був випущений у 2001 році. Тепер Qt був доступний у Windows, Mac OS X, Unix та Linux (настільний та вбудований). Qt 3 забезпечив 42 нові класи, і його код перевищив 500 000 рядків. Qt 3 був головним кроком вперед від Qt 2, включаючи значно вдосконалену підтримку локалі та Unicode, абсолютно новий віджет для перегляду та редагування тексту та регулярний клас вираження регулярних виразів. У 2002 році Qt 3 виграв Software Development Times «Jolt Productivity Award».

Також у 2005 році Trolltech відкрив представництво в Пекіні, щоб надати клієнтам у Китаї та регіоні послуги з продажу, навчання та технічну підтримку для Qtopia.

З моменту народження Trolltech популярність Qt зросла і продовжує зростати донині. Цей успіх є відображенням як якості Qt, так і того, наскільки приємно використовувати цей фреймворк. В останнє десятиліття Qt перейшло від продукту, яким користуються декілька вибраних «знаю», до того, яким щодня користуються тисячі клієнтів і десятки тисяч розробників з відкритим кодом у всьому світі [17].

PyQt – це бібліотека, яка дозволяє використовувати Qt GUI на мові Python. Використовуючи його з Python, ви можете створювати додатки набагато швидше, не втрачаючи при цьому великої швидкості C ++.

PyQt5 практично повністю реалізує можливості Qt. А це понад 600 класів, більше 6000 функцій і методів [3].

PyQt5 також включає в себе Qt Designer (Qt Creator) – дизайнер графічного інтерфейсу користувача. Програма pyuic генерує Python код з файлів, створених в Qt Designer.

Це робить PyQt5 дуже корисним інструментом для швидкого прототипування. Крім того, можна додавати нові графічні елементи управління, написані на Python, в Qt Designer (рисунок 4.9).

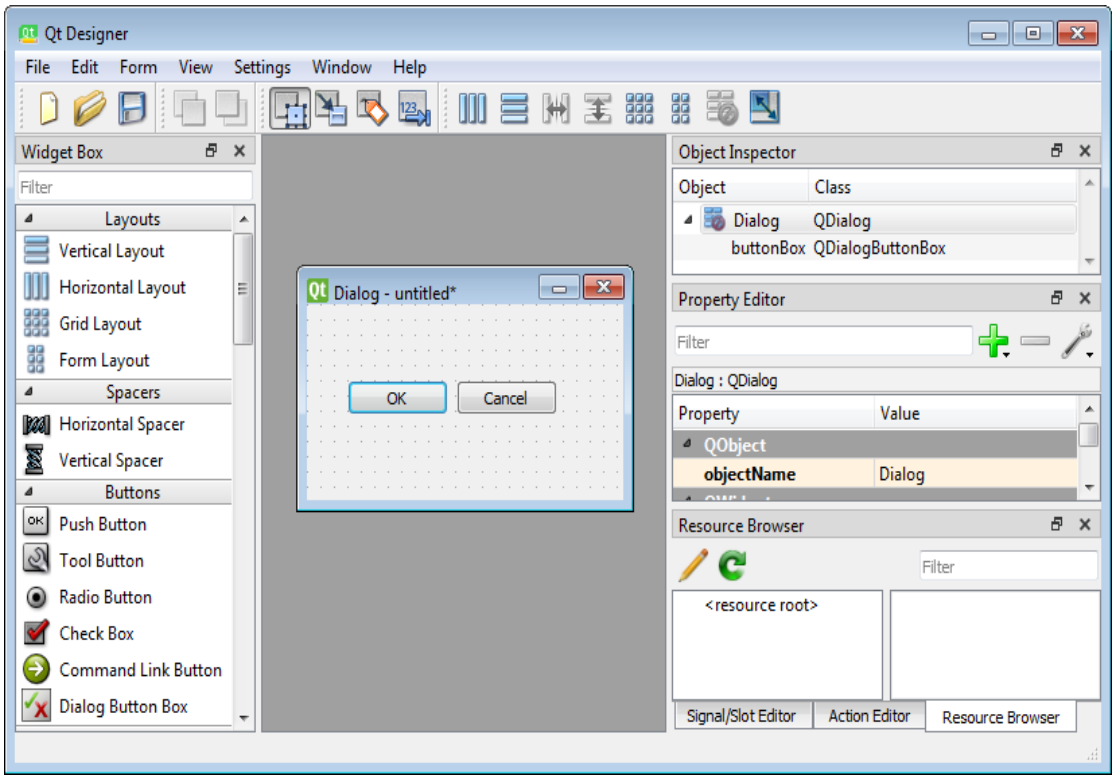

Рисунок 4.9 – Qt Designer

Раніше PyQt5 поставлявся разом із середовищем розробки Eric, яке написане на PyQt5. Eric має вбудований відладчик і може бути використана

для створення консольних програм. Тепер вона доступна в якості окремого проекту.

Іншим популярним засобом розробки GUI мовою Python є Tkinter. І Tkinter, і PyQt є корисними для проектування графічних інтерфейсів, але в той же час вони відрізняються з точки зору пристосованості та функціональності.

Здебільшого, Tkinter – про те, щоб самостійно писати графічний інтерфейс, запрограмувати налаштування чи функціональність у тому ж сценарії.

З іншого боку, в PyQt ви відокремлюєте графічний інтерфейс у сценарії та використовуєте свої знання Python від іншого сценарію. Тобто, як вже було сказано вище, в PyQt є засіб QtDesigner для створення дизайну, який потім можна інтегрувати у власну систму. В Tkinter такий засіб відсутній. Тож, у порівнянні з Tkinter плюсами використання PyQt є наступне:

– гнучкість кодування – програмування GUI з Qt розроблено навколо концепції сигналів і слотів для встановлення зв'язку між об'єктами. Це дозволяє гнучкість під час роботи з подіями графічного інтерфейсу і призводить до більш гладкої кодової бази;

– більше ніж фреймворк – Qt використовує широкий набір власних API;

– платформ для створення мереж, створення бази даних та багато іншого. Він пропонує первинний доступ до них через унікальний API;

– різні компоненти інтерфейсу користувача – Qt пропонує декілька віджетів, таких як кнопки або меню, розроблені з базовим виглядом на всіх підтримуваних платформах;

– різні ресурси навчання – оскільки PyQt є одним з найбільш використовуваних фреймворків інтерфейсу для Python, ви можете отримати простий доступ до широкого набору документації.

Але, оскільки це в першу чергу фреймворк для C++, то з іншого боку він має і недоліки:

– відсутність специфічної для Python документації для класів в PyQt5;

– це вимагає багато часу для розуміння всіх деталей PyQt, це означає, що це досить важка крива навчання.

Переваги використання Tkinter:

– доступно безкоштовно для комерційного використання;

– він розміщений у базовій бібліотеці Python;

– створення виконуваних файлів для додатків Tkinter є більш доступним, оскільки Tkinter включений в Python, і, як наслідок, він не має інших залежностей. Зм. Арк. № докум. Підпис Дата Арк. 35 ІАЛЦ.045430.003 ПЗ;

– простий для розуміння та оволодіння.

Tkinter – це обмежена бібліотека з простим API, що є основним вибором для швидкого створення простих графічних інтерфейсів для скриптів Python. Та очевидні недоліки його використання:

– Tkinter не включає в себе складні віджети;

– він не має подібного інструменту, як Qt Designer для Tkinter;

– дизайн виглядає набагато гірше (рисунок 4.10).

Нарешті, останнім популярним засобом розробки GUI мовою Python є Kivy. Kivy – бібліотека Python, призначена для створення мультимедійних додатків, що підтримують мультимедіа. Мета – дозволити швидке та просте проектування взаємодії, а також швидке прототипування, багаторазовий код та функції розгортання.

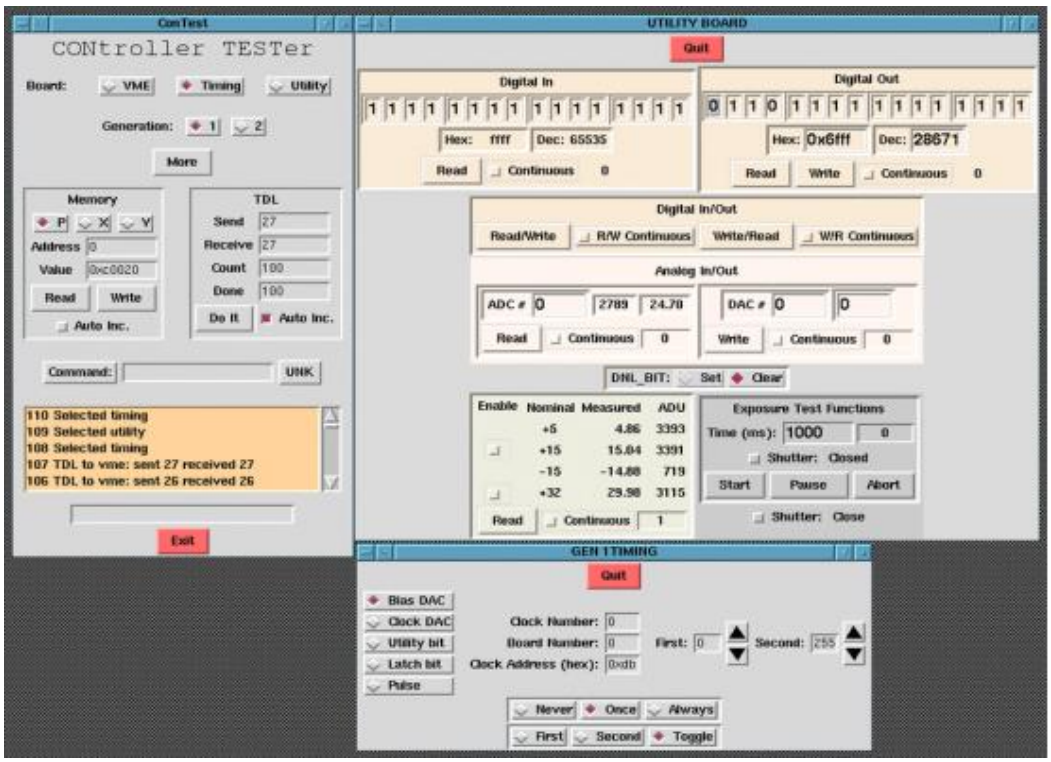

Рисунок 4.10 – Дизайн GUI з використанням Tkinter

Kivy розроблений на базі OpenGL, і його можна використовувати з декількома пристроями введення. Коли PyQt представляє собою повноцінний фреймворк для розробки графічного інтерфейсу, Kivy – більше використовується для міжплатформових засобів мобільної розробки.

Kivy працює безперебійно на Linux, Windows, OS X, Android та Raspberry Pi. І значна перевага використання Kivy os в тому, що ви можете запускати один і той же код на всіх підтримуваних платформах.

Все це досягається у вигляді дерева, який є дещо більш точним та організованим у порівнянні з іншими інструментами для програмування.

Оскільки Kivy більш орієнтований під мобільну розробку, в ньому також немає комплексних речей, складних віджетів тощо. Приклад додатку написаного за допомогою Kivy зображено на рисунку 4.11.

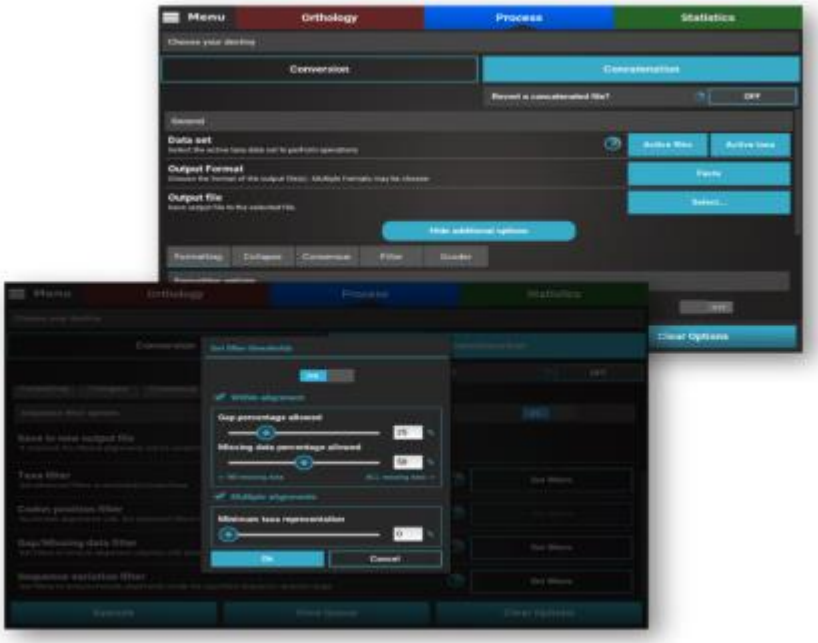

Рисунок 4.11 – Дизайн GUI з використанням Kivy

Я обрав PyQt тому що він є відмінним засобом для створення графічного інтерфейсу для голосового помічника з декількох причин.

По-перше, PyQt є потужною бібліотекою, яка надає розширені можливості для розробки графічного інтерфейсу. Вона має широкий набір компонентів і інструментів, що дозволяє створювати багатофункціональні та візуально привабливі інтерфейси. Також, вона базується на популярній бібліотеці Qt, що надає переваги в плані кросплатформенності. Графічні інтерфейси, створені з його використанням, можуть працювати на різних операційних системах, таких як Windows, macOS і Linux, забезпечуючи єдиний досвід користувача для всіх платформ.

Що не мало важливо, PyQt має добре документовану і активну спільноту. Це означає, що є багато ресурсів, які допомагають розробникам вивчити бібліотеку та отримати підтримку у разі потреби. Існує безліч прикладів коду, документація, форуми та спеціалізовані групи, де можна знайти відповіді на питання та поради щодо розробки графічного інтерфейсу з використанням PyQt.

Загалом, PyQt є потужним інструментом, який дозволяє розробникам створювати графічний інтерфейс для голосового помічника з високою ефективністю і гнучкістю. Він надає доступ до багатого набору функціональності та забезпечує кросплатформенну підтримку, що робить його прекрасним вибором для розробки графічного інтерфейсу для голосових асистентів.

Отже, ми побачили що найкращим засобом для розробки графічного інтерфейсу є PyQt, так як в Tkinter неможливо зробити повноцінний комплексний додаток за відсутності складних віджетів, а Kivy більш орієнтований під мобільні додатки.

ВИСНОВКИ

В результаті виконання задач, поставлених в рамках кваліфікаційної роботи, були систематизовані та удосконаленні теоретичні знання та практичні навички, було проведено аналіз методів побудови баз знань і розроблено прототип програми.

Був проведений збір та глибокий аналіз матеріалів, що стосуються теми кваліфікаційної роботи – «Голосовий асистент користувача на базі нейромережевих технологій». Матеріали були систематизовані та проаналізовані, що дозволило отримати більш ясну картину як задачі в цілому, так і специфічних особливостей запропонованого підходу.

Для перевірки методу був розроблений програмний прототип з використанням мов програмування Python та з функціоналом пошуку, отримання інформації щодо погоди та часу, надсилання електронних повідомлень та інше.

Також були сформовані основні задачі майбутньої програми яка має вирішити основні проблема в цьому напрямку, та описані основна ідея та математичне формулювання запропонованого до дослідження методу.

Під час проведення дослідження, були також здобуті практичні знання та навички у організаційно-управлінській роботі, навички з прийняття самостійних організаційно-технічних та організаційно-управлінських рішень у конкретних виробничих умовах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fogel, D. B., Owens, A. J., & Walsh, M. J. (2015). Artificial Intelligence through Search. Springer.

2. PyQt5 URL : https://ru.wikipedia.org/wiki/PyQt (дата звернення 15.04.2023).

3. QtDesigner URL: https://ru.wikipedia.org/wiki/Qt_Designer (дата звернення 15.04.2023).

4. DesignerSwaroop C.H. «A Byte of Python».

5. Python URL: https://ru.wikipedia.org/wiki/Python (дата звернення 15.04.2023).

6. Challenges in benchmarking stream learning algorithms with real-world data / V. M. A. Souza et al. Data mining and knowledge discovery. 2020. Vol. 34, no. 6. P. 1805–1858. URL: https://doi.org/10.1007/s10618-020-00698-5 (дата звернення 15.04.2023).

7. Lucas, S. M. (2019). Artificial intelligence in games: A textbook. CRC Press.

8. R. Fielding, Architectural styles and the design of network-based software architectures, PhD Thesis. University of California, Irvine, Information and Computer Science Department. 2000.

9. Witten, Ian H.; Frank, Eibe; Hall, Mark A. (30 January 2011). Data Mining: Practical Machine Learning Tools and Techniques (3 ed.). Elsevier.

10. Документація бази даних «MongoDB» URL: https://www.mongodb.com/ (дата звернення 15.04.2023).

11. What is Apple Siri URL: https://www.pocketlint.com/apps/news/apple/112346-what-is-siri-apple-s-personal-voice-assistant-explained (дата звернення 15.04.2023).

12. Long short-term memory recurrent neural network architectures for large scale acoustic modeling (2014), Hasim Sak et al., Google Research .

13. Wav2Letter: an End-to-End ConvNet-based Speech Recognition System (2016), Ronan Collobert et al., Facebook AI Research.

14. Indurkhya, N., and Damerau, F. (2010). Handbook Of Natural Language Processing, 2nd Edition. Boca Raton, FL: CRC Press.

15. . Nadeau D. A survey of named entity recognition and classification / D. Nadeau, S. Sekine // National Research Council Canada – 2006.

16. Li J. A Survey on Deep Learning for Named Entity Recognition / Jing Li et al. // URL: https://arxiv.org/pdf/1812.09449.pdf (дата звернення 15.04.2023).

17. Coursera. Рекурентні нейронні мережі URL: https://www.coursera.org/learn/nlp-sequence-models/lecture/ftkzt/recurrentneural-network-model (дата звернення 15.04.2023).

18. QT story URL: https://rtime.felk.cvut.cz/osp/prednasky/gui/the-qt-story/ (дата звернення 15.04.2023).

19. Голосовий асистент «Amazon Alexa» URL: https://developer.amazon.com/en-US/docs/alexa/alexa-voiceservice/api-overview.html (дата звернення 15.04.2023).

20. Голосовий асистент «Siri» URL: https://developer.apple.com/siri/ (дата звернення 15.04.2023).

21. Голосовий асистент «Wit AI» URL: https://wit.ai (дата звернення 15.04.2023).

22. Віртуальний помічник [URL: https://uk.wikipedia.org/wiki/Віртуальний_помічник (дата звернення 15.04.2023).

23. Голосовий асистент «MyCroft» URL: https://mycroft.ai/ (дата звернення 15.04.2023).

24. Распознавание по голосу URL: https://ru.wikipedia.org/wiki/Распознавание_по_голосу (дата звернення

15.04.2023).

25. Лингвистический процессор. Предварительная обработка текста. Пофразовая обработка текста. URL: https://studfile.net/preview/1401101/page:48/ (дата звернення 15.04.2023).

26. H. McWilliam, Analysis tool web services, from the EMBL-EBI, Nucleic Acids Res., 2013, vol. 41 (pg. W597-W600).

27. R. Fielding, Architectural styles and the design of network-based software architectures, PhD Thesis. University of California, Irvine, Information and Computer Science Department. 2000.

28. What is Google Assistant URL: https://www.pocket-lint.com/apps/news/google/137722-what-is-google-assistant-howdoes-it-work-and-which-devices-offer-it (дата звернення 15.04.2023).

29. Голосове управління комп'ютером на основі глосарію за допомогою алгоритмів розпізнавання мови URL: https://ela.kpi.ua/bitstream/123456789/34539/1/Dudchenko_bakalavr.pdf (дата звернення 15.04.2023).

30. Голосовий асистент «Cortrana» URL: https://docs.microsoft.com/en-us/cortana/skills/. PyQt5 URL: https://ru.wikipedia.org/wiki/PyQt (дата звернення 15.04.2023).

31. 2. Dalkir, K. (2017). Knowledge management in theory and practice. MIT press.

32. Critical study of neural networks in detecting intrusions URL: https://www.sciencedirect.com/science/article/abs/pii/S0167404808000357 (дата звернення 15.04.2023).

33. The basic ideas in neural networks URL: https://go.gale.com/ (дата звернення 15.04.2023).

34. Singing voice synthesis based on deep neural networks URL: https://www.isca-speech.org/archive_v0/Interspeech_2016/pdfs/1027.PDF (дата звернення 15.04.2023).

35. Sinsy: A Deep Neural Network-Based Singing Voice Synthesis System URL: https://ieeexplore.ieee.org/abstract/document/9511835 (дата звернення 15.04.2023).

36. Singing voice synthesis based on convolutional neural networks URL: https://arxiv.org/abs/1904.06868 (дата звернення 15.04.2023).

37. Review of Neural Networks for Speech Recognition URL: https://ieeexplore.ieee.org/abstract/document/6796621 (дата звернення 15.04.2023).

38. Review of Neural Networks for Speech Recognition URL: https://ieeexplore.ieee.org/abstract/document/6796621 (дата звернення 15.04.2023).

39. Speech recognition using neural networks URL: https://www.proquest.com/openview (дата звернення 15.04.2023).

40. Models URL: https://alphacephei.com/vosk/models (дата звернення 15.04.2023).

41. Синтез речи URL: https://neerc.ifmo.ru/wiki/index.php?title=Синтез_речи (дата звернення 15.04.2023).

ДОДАТОК А

Код програми

Main.py

```
from sympy.physics.units import temperature


import config
import requests
import stt
import tts
from fuzzywuzzy import fuzz
import datetime
from num2words import num2words
import webbrowser
import random
import webbrowser



print(f»{config.VA_NAME} (v{config.VA_VER}) начал свою работу ...»)



def va_respond(voice: str):
    print(voice)
    if voice.startswith(config.VA_ALIAS):
        # обращаются к ассистенту
        cmd = recognize_cmd(filter_cmd(voice))

        if cmd['cmd'] not in config.VA_CMD_LIST.keys():
            tts.va_speak(«Что?»)
        else:
```

```
            execute_cmd(cmd['cmd'])


def filter_cmd(raw_voice: str):
    cmd = raw_voice

    for x in config.VA_ALIAS:
        cmd = cmd.replace(x, «»).strip()

    for x in config.VA_TBR:
        cmd = cmd.replace(x, «»).strip()

    return cmd


def recognize_cmd(cmd: str):
    rc = {'cmd': '', 'percent': 0}
    for c, v in config.VA_CMD_LIST.items():

        for x in v:
            vrt = fuzz.ratio(cmd, x)
            if vrt > rc['percent']:
                rc['cmd'] = c
                rc['percent'] = vrt

    return rc


def execute_cmd(cmd: str):
    if cmd == 'help':
```

```
        # help
        text = «Я умею: ...»
        text += «произносить время ...»
        text += «рассказывать анекдоты ...»
        text += «открывать браузер...»
        text += «смотреть погоду...»
        text += «искать в интернете...»
        text += «и отправлять сообщения...»
        tts.va_speak(text)
        pass
    elif cmd == 'ctime':
        # current time
        now = datetime.datetime.now()
        text = «Сейч+ас « + num2words(now.hour, lang='ru') + « « +
num2words(now.minute, lang='ru')
        tts.va_speak(text)


    elif cmd == 'joke':
        jokes = ['Как смеются программисты? ... exe exe exe',
                 'ЭсКьюЭль запрос заходит в бар, подходит к двум
столам и спрашивает .. «м+ожно присоединиться?»',
                 'Программист это машина для преобразования кофе в
код']


        tts.va_speak(random.choice(jokes))


    elif cmd == 'weather':
            api_key = 'YOUR_API_KEY'
            city = 'YOUR_CITY'
```

```
        url =
f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid=
{api_key}'
        response = requests.get(url)
        data = response.json()
        if data['cod'] == 200:
            weather = data['weather'][0]['description']
            text = f»У місті {city} зараз {weather}. Температура
становить {temperature} градусів Цельсія.»
        tts.va_speak(text)


    elif cmd == 'open_browser':
        chrome_path = 'C:\Program
Files\Google\Chrome\Application/chrome.exe %s'
        webbrowser.get(chrome_path).open(«http://python.org»)


def search_web(query: str):
        if 'google' in query.lower():
         search_google(query)


def search_google(query: str):
         search_url = «https://www.google.com/search?q=«
         encoded_query = query.replace(' ', '+')
         url = search_url + encoded_query
         webbrowser.open(url)


search_web('search on Google')


def send_message(recipient: str, message: str, method: str = 'sms'):
        if method == 'sms':
            send_sms(recipient, message)
```

```
        elif method == 'email':
            send_email(recipient, message)
        else:
            print(f»Unsupported method: {method}»)

def send_sms(recipient: str, message: str):
        # Код для надсилання SMS повідомлення
        print(f»Sending SMS to {recipient}: {message}»)

def send_email(recipient: str, message: str):
        # Код для надсилання електронної пошти
        print(f»Sending email to {recipient}: {message}»)

# начать прослушивание команд
stt.va_listen(va_respond)
```

stt.py

```
import vosk
import sys
import sounddevice as sd
import queue
import json

model = vosk.Model(«model»)
samplerate = 16000
device = 2

q = queue.Queue()
```

```
def q_callback(indata, frames, time, status):
    if status:
        print(status, file=sys.stderr)
    q.put(bytes(indata))



def va_listen(callback):
    with sd.RawInputStream(samplerate=samplerate,
blocksize=8000, device=device, dtype='int16',
                           channels=1, callback=q_callback):


        rec = vosk.KaldiRecognizer(model, samplerate)
        while True:
            data = q.get()
            if rec.AcceptWaveform(data):
                callback(json.loads(rec.Result())[«text»])
            #else:
            #    print(rec.PartialResult())
```

stt.py(2)

```
import torch
import sounddevice as sd
import speech_recognition as sr
import time
import numpy
from glob import glob

device = torch.device('cpu')
model, decoder, utils = torch.hub.load(repo_or_dir='snakers4/silero-
models',
```

```
                                model='silero_stt',
                                language='en', # en, ru
                                device=device)
(read_batch, split_into_batches,
 read_audio, prepare_model_input) = utils

def callback(_r, audio):
    try:
        # CONVERT raw wav data to NumPy array
        # wav_raw = audio.get_wav_data()
        # data_s16 = numpy.frombuffer(wav_raw,
dtype=numpy.int16, count=len(wav_raw) // 2, offset=0)
        # np_audio = data_s16 * 0.5 ** 15

        # Play it via sounddevice
        #sd.play(np_audio, m.SAMPLE_RATE)
        #time.sleep(len(np_audio) / m.SAMPLE_RATE)
        #sd.stop()

        print(«Распознание ...»)

        with open('speech.wav', 'wb') as f:
            f.write(audio.get_wav_data())

        test_files = glob('speech.wav')
        batches = split_into_batches(test_files, batch_size=10)
        input = prepare_model_input(read_batch(batches[0]),
                                    device=device)

        output = model(input)
```

```
        for example in output:
            print(decoder(example.cpu()))


        # voice = recognizer.recognize_google(audio, language=«ru-
RU»).lower()
        # print(«[log] Распознано: « + voice)


    except sr.UnknownValueError:
        print(«[log] Голос не распознан!»)



# запуск
r = sr.Recognizer()
r.pause_threshold = 0.5
m = sr.Microphone(device_index=1)

with m as source:
    r.adjust_for_ambient_noise(source)

stop_listening = r.listen_in_background(m, callback)
while True: time.sleep(0.1)
```

tts.py

```
import torch
import sounddevice as sd
import time

language = 'ru'
```

```
model_id = 'ru_v3'
sample_rate = 48000 # 48000
speaker = 'baya' # aidar, baya, kseniya, xenia, random
put_accent = True
put_yo = True
device = torch.device('cpu') # cpu или gpu
text = «Привет!!!»

model, _ = torch.hub.load(repo_or_dir='snakers4/silero-models',
                          model='silero_tts',
                          language=language,
                          speaker=model_id)
model.to(device)



# воспроизводим
def va_speak(what: str):
    audio = model.apply_tts(text=what+»..»,
                            speaker=speaker,
                            sample_rate=sample_rate,
                            put_accent=put_accent,
                            put_yo=put_yo)

    sd.play(audio, sample_rate * 1.05)
    time.sleep((len(audio) / sample_rate) + 0.5)
    sd.stop()

# sd.play(audio, sample_rate)
# time.sleep(len(audio) / sample_rate)
# sd.stop()
```

config.py

```
VA_NAME = 'Помощник'

VA_VER = «2.0»

VA_ALIAS = ('помощник', 'асистент', 'помоги', 'друг')

VA_TBR = ('скажи', 'покажи', 'ответь', 'произнеси', 'расскажи', 'сколько')

VA_CMD_LIST = {
   «help»: ('список команд', 'команды', 'что ты умеешь', 'твои навыки',
'навыки'),
   «ctime»: ('время', 'текущее время', 'сейчас времени', 'который час'),
   «joke»: ('расскажи анекдот', 'рассмеши', 'шутка', 'расскажи шутку',
'пошути', 'развесели'),
   «open_browser»: ('открой браузер', 'запусти браузер', 'открой гугл хром',
'гугл хром'),
   «weather»: ('погода', 'как там на улице', 'сколько градусов за окном'),
   «searh_web»: ('найди', 'посмотри в интернете', 'найди в интернете',
'напиши в интернете'),
   «send_message»: ('напиши смс', 'отправь сообщение', 'напиши
сообщение')
}
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	73 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Липенко А.В.			Голосовий асистент користувача на базі нейромережевих технологій	Шифр групи	Код напр./спец.
Перевірив		Єрохін А.Л,				СШІм-21-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					