МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА СУПРОВОДЖЕННЯ КОРИСТУВАЧА ПК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувач _____ Іван СКОПЕНКО

«___»_____ 2024 р.

Керівник канд. пед. наук, доцент _____ Надія БОЛЮБАШ

«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Факультет комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Скопенко Івана Вікторовича_____
((прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система супроводження користувача ПК на основі нейромережевих моделей».

Керівник роботи: Болюбаш Надія Миколаївна, доцент кафедри інтелектуальних інформаційних систем, канд. пед. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «03» червня 2024 р. № 140/1.

2. Строк представлення кваліфікаційної роботи «___» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: інтелектуальна система супроводження користувача, яка дозволяє виконувати команди користувальницького інтерфейсу за рахунок голосових команд в операційній системі Windows на українській мові; нейромережеві моделі та методи машинного навчання й обробки природної мови для розуміння команд користувача.

4. Перелік питань, що підлягають розробці: дослідження теоретичних засад супроводження користувача ПК на основі нейромережевих моделей і методів обробки природної мови та здійснення аналізу існуючих рішень; обґрунтування вибору технологій і засобів розробки системи підтримки користувача ПК; проєктування та здійснення програмної реалізації інтелектуальної системи супроводження користувача ПК у вигляді голосового асистента.

5. Перелік графічних матеріалів: презентація, рисунки, таблиці.

Керівник роботи _____ Надія БОЛЮБАШ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Іван СКОПЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «07» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система супроводження користувача ПК на основі нейромережевих моделей

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	03.06.2024	07.06.2024	Виконано
2	Аналіз предметної області та постановка задачі	10.06.2024	20.06.2024	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, аналіз публікацій та існуючих підходів до гоосової підтримки користувача ПК	21.06.2024	01.07.2024	Виконано
4	Огляд існуючих моделей та методів машинного навчання й обробки природної мови та вибір стеку технологій розробки	01.09.2024	25.10.2024	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	26.10.2024	21.11.2024	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	22.11.2024	22.11.2024	Виконано
7	Корегування роботи за результатами попереднього захисту	23.11.2024	05.12.2024	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	06.12.2024	06.12.2024	Виконано
9	Доробка та остаточне оформлення КР	07.12.2024	10.02.2024	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2024	17.12.2024	Виконано

Керівник роботи _____ Надія БОЛЮБАШ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Іван СКОПЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«19» червня 2024 р.

АНОТАЦІЯ

на кваліфікаційну роботу

Скопенко Івана Вікторовича

на тему: «**ІНТЕЛЕКТУАЛЬНА СИСТЕМА СУПРОВОДЖЕННЯ КОРИСТУВАЧА ПК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ**»

Кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації інтелектуальної системи, яка підтримує аудіоспілкування з користувачем українською мовою та реалізує виконання команд користувальницького інтерфейсу в операційній системі Windows. Що є актуальним в умовах високих темпів інформатизації, оскільки дозволяє автоматизувати виконання рутинних завдань роботи з комп'ютером у більш звичному для людини форматі.

Об'єкт дослідження – процес виконання команд користувальницького інтерфейсу в операційній системі ПК.

Предмет дослідження – програмні засоби та нейромережеві моделі для голосового спілкування з користувачем по виконання команд UI-інтерфейсу в операційній системі Windows.

Мета дослідження – підвищення ефективності взаємодії користувача з операційною системою ПК шляхом створення інтелектуальної системи із використанням методів обробки природної української мови на основі нейромережевих моделей.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та додатків. У першому розділі розкрито теоретичні засади супроводження користувача ПК на основі нейромережевих моделей і методів обробки природної мови та здійснення аналізу існуючих рішень. У другому розділі обґрунтовано вибір технологій і засобів розробки системи підтримки користувача ПК. У третьому розділі описано проєктування, здійснення програмної реалізації інтелектуальної системи супроводження користувача ПК у вигляді голосового асистента на українській мові та оцінку її ефективності.

Кваліфікаційна робота містить 76 сторінок, 40 рисунків, 1 таблицю, 26 джерел.

Ключові слова: *інтерфейс користувача, обробка природної мови, велика мовна модель, трансформер, токенізація*.

ABSTRACT

of the qualification work

Skopenko Ivana Viktorovycha

on the subject: «**INTELLIGENT PC USER SUPPORT SYSTEM BASED ON NEURAL NETWORK MODELS**»

The master's qualification work is devoted to the development and of software implementation of an intelligent system that supports audio communication with the user in Ukrainian and implements the execution of user interface commands in the Windows operating system. Which is relevant in conditions of high rates of informatization, as it allows you to automate the performance of routine tasks of working with a computer in a format more familiar to humans.

Object of research – the process of executing user interface commands in a PC operating system.

Subject of research – software tools and neural network models for voice communication with the user to execute user interface commands in the Windows operating system.

The purpose of the study is to increasing the efficiency of user interaction with the PC operating system by creating an intelligent system using natural Ukrainian language processing methods based on neural network models.

The master's qualification work consists of an introduction, three sections, conclusions and appendices. The first section reveals the theoretical principles of PC user support based on neural network models and natural language processing methods and the analysis of existing solutions. The second section justifies the choice of technologies and tools for developing a PC user support system. The third section describes the design, software implementation of an intelligent support system for a PC in the form of a voice assistant in the Ukrainian language and evaluation of its effectiveness.

The master’s thesis contains 76 page, 40 figures, 1 table, 26 sources.

Key words: *user interface, natural language processing, large language model, transformer, tokenization*.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПК – персональний комп’ютер

ШІ – Штучний інтелект

AI – Artificial Intelligence

GPT – Generative Pre-trained Transformer

GTTS – Google Text-to-Speech

LlaMA – Large Language Model Meta AI

LLM – Large Language Model

NLP – Natural Language Processing

PyCaw – Python Core Audio Windows

RAG – Retrieval Augmentation Generation

RNN – Recurrent Neural Networks

UI – User Interface

ВСТУП

Актуальність. Високі темпи інформатизації сучасного суспільства обумовлюють впровадження засобів автоматизації UI-інтерфейсу користувачів ПК. Перспективним напрямком підвищення ефективності взаємодії користувачів з операційною системою є використання нейромережевих моделей для підтримки аудіоспілкування з ними, яке може автоматизувати виконання основних команд користувальницького інтерфейсу. Для осіб із обмеженими фізичними можливостями та недостатньою інформаційною підготовкою це сприяє прискоренню їх інтеграції у сучасне суспільство.

Що є актуальним для України, яка через тривалий військовий конфлікт має велику кількість постраждалих людей, які потребують соціальної та професійної адаптації. Тому існує потреба в розробці інтелектуальної системи супроводження користувача, яка б допомогла полегшити взаємодію з ПК, автоматизували рутинні завдання та забезпечили підтримку у роботі з інформаційними технологіями. Особливо важливим є застосування методів обробки природної мови (англ. Natural Language Processing, NLP) на основі нейромережевих моделей, здатних адаптуватися до індивідуальних потреб кожного користувача.

В основі сучасних систем NLP лежать великі мовні моделі, архітектура яких базується на трансформерах – великих наборах нейронних мереж із можливістю самонавчання. Впровадження трансформерів стало ключовим проривом у обробці природної мови, їх архітектура дозволяє використовувати дуже великі нейронні моделі з сотнями мільярдів параметрів і підтримкою багатьох мов. До найбільш популярних LLM належать GPT-3 і GPT-4 від OpenAI, BERT і PaLM від Google, модель Granite від IBM, модель Jurassic-1 від AI21 Labs а також модель LLAMA, створена Meta AI.

До реалій сьогодення відносять також розробку голосових асистентів провідними IT-компаніями Apple, Google, Microsoft та Amazon. Це такі голосові помічники користувачів, як Siri, Google Assistant, Amazon Alexa, Microsoft Cortana.

Більшість віртуальних помічників є безкоштовними та інтегрованими в пристрої користувачів, однак їх функціональність по виконанню команд UI-інтерфейсу є обмеженою. Також відсутня підтримка спілкування українською мовою. Голосовий помічник Windows 11 Copilot виконує багато команд користувальницького інтерфейсу по управлінню ПК, однак все це працює тільки на англійській мові. Це обумовлює потребу у розробці інтелектуальної системи супроводження користувача ПК, яка базується на нейромережевих моделях, здатна забезпечити підтримку шляхом автоматизації виконання рутинних операцій і взаємодії з програмним забезпеченням по голосовим командам користувача, адаптуючись до його специфічних потреб.

Метою дослідження є підвищення ефективності взаємодії користувача з операційною системою ПК шляхом створення інтелектуальної системи із використанням методів обробки природної української мови на основі нейромережевих моделей.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

– дослідити теоретичні засади голосового спілкування з користувачем по виконанню команд UI-інтерфейсу ОС Windows із використанням нейромережевих моделей і методів обробки природної мови та здійснити аналіз існуючих рішень;

– обґрунтувати вибір технологій та засобів розробки системи підтримки користувача ПК;

– здійснити проєктування, програмну реалізацію інтелектуальної системи супроводження користувача ПК у вигляді голосового асистента з підтримкою української мови та оцінити її ефективність.

Об’єктом дослідження є процес виконання команд користувальницького інтерфейсу в операційній системі ПК.

Предметом дослідження є програмні засоби та нейромережеві моделі для голосового спілкування з користувачем по виконання команд UI-інтерфейсу в операційній системі Windows.

Методологічною основою дослідження є загальнонаукові аналітичні методи, нейромережеві моделі обробки природної мови, методи машинного навчання, які дозволили вивчити предмет та об’єкт дослідження, дослідити розвиток науково-методичних засад, напрямів та шляхів підвищення ефективності взаємодії користувача ПК з операційною системою шляхом підтримки аудіоспілкування з ним українською мовою.

Результати дослідження обговорювалися на Всеукраїнській науково-практичній конференції «Інтелектуальні інформаційні системи» (2-4 грудня 2024 року) та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що розроблена інтелектуальна система може бути використана для автоматизації виконання рутинних завдань роботи з комп’ютерною технікою особами із недостатньою інформаційною підготовкою й обмеженими фізичними можливостями, що сприятиме прискоренню їх інтеграції у сучасне суспільство.

Структура кваліфікаційної роботи. Відповідно до мети, завдань і предмета дослідження кваліфікаційна робота складається із вступу, трьох розділів, висновку та списку використаних джерел. Загальний обсяг кваліфікаційної роботи – 76 сторінок. Кількість використаних джерел – 26.

1 ТЕОРЕТИЧНІ ЗАСАДИ ПІДТРИМКИ UI-ІНТЕРФЕЙСА ОС WINDOWS НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ОБРОБКИ ПРИРОДНОЇ МОВИ

1.1 Нейромережеві моделі обробки природної мови

Обробка природної мови (англ. Natural Language Processing, NLP) – це набір методів, які допомагають комп'ютерній системі приймати людську мову [1]. NLP є підрозділом штучного інтелекту та однією з його найскладніших задач, яка не вирішена в повній мірі до цих пір.

Вперше про це зайшла мова у 1950-ті роки, коли відомий англійський учений Алан Тьюринг опублікував статтю «Обчислювальні машини і розум», у якій запропонував так званий «Тест Тьюрінга». Одним із його критеріїв є здатність машини автоматично інтерпретувати та генерувати людську мову. Перші спроби створення систем такого роду були не дуже вдалими. У 1966 році Джозеф Вейценбаум у стінах Массачусетського технологічного інституту створив перший у світі чат-бот «Еліза», який вів діалог, використовуючи техніку активного слухання. Проте програма була далека від розуміння суті діалогу.

Для відповіді на питання розуміння сутності необхідно з'ясувати, як природна мова використовується людьми. Коли людина чує або читає якусь фразу, в її підсвідомості відбувається одночасно кілька процесів: сприйняття, розуміння сутності та реагування. Сприйняття є процесом перекладу сенсорного сигналу в символьний формат. Розуміння сутності – це сама складна задача, з якою не завжди справляються навіть люди зі своїм природним інтелектом. Із-за незнання контексту та неправильної інтерпретації фрази можуть виникнути різні плутанини, а іноді конфліктні ситуації. Саме тому дуже важливо правильно сприймати сутність слова, контекст сказаного або написаного. Реакція є результатом прийняття рішення. Це досить проста задача, яка вимагає формування набору можливих відповідей на основі сутності сприйнятої фрази, контексту.

Алгоритми обробки природного мови працюють за таким же принципом. Сприйняття – це процес перекладу вхідної інформації в зрозумілий для комп'ютера набір символів. Якщо це текст у чаті, то такий вхідний набір буде прямим. Якщо це аудіофайл або рукописний текст, то для початку його потрібно перевести в зручний формат. З цим успішно справляються сучасні нейронні мережі.

Задачу реагування на текст також успішно вирішили шляхом введення альтернативи та порівняння результатів один із одним. Для чат-бота це може бути текстова відповідь з його бази знань, а для голосового помічника у операційній системі – виконання якоїсь дії користувальницького інтерфейсу.

Для розумінням сутності нині розповсюдженими є такі види аналізу: статистичний, формально-граматичний та нейромережевий.

Упродовж десятиліть після створення «Елізи» проривів у сфері NLP не спостерігалося до виникнення перших алгоритмів машинного навчання в 1980 роках. У цей же час з'явилися системи статистичного машинного перекладу, завдяки чому дослідження відновилися. Статистика широко застосовується в сервісах машинного перекладу, автоматичних рецензентах і деяких чат-ботах. Суть методу полягає в наданні моделі великої кількості масиву текстів, в яких встановлені статистичні закономірності. Потом такі моделі використовуються для перекладу текстів або генерації нових, іноді з певним розумінням контексту.

Формально-граматичний підхід являє собою математичний апарат, який дозволяє точно й однозначно визначати значення фрази на природній мові настільки, наскільки це можливо для комп'ютера. Однак це не завжди вдається зробити. Для розвинутих мов точний і детальний опис слова в математичних термінах є надзвичайно складною проблемою. Тому формально-граматичний підхід частіше використовується для синтаксичного аналізу штучних мов, з яких спеціально видаляють неоднозначності при проєктуванні.

Роботи в області обробки мови різко зросли в 2010-ті роки. У цей період з'явилося багато розробок із використанням нейронних мереж для обробки природної мови, якими користуються і сьогодні, таких як чат-боти, автокоректори,

голосові помічники тощо. У нейромережевому підході для розпізнавання сутності вхідної фрази і генерації реакцій системи ШІ використовуються нейронні мережі глибокого навчання. Вони навчаються на парах стимул-реакцій, де стимулом є фраза на природній мові, а реакція – це відповідь на тій же мові або виконання якоїсь дії системою ШІ. Цей підхід виявився дуже перспективним. Серед найбільш широко застосовуваних нейронних мереж для обробки природної мови стали використовувати рекурентні нейронні мережі (англ. Recurrent Neural Networks, RNN), рекурсивні нейронні мережі (англ. Recursive Neural Networks, RNN) та згорткові нейронні мережі (англ. Convolutional Neural Networks, CNN).

Наступна революція в системах обробки природної мови відбулася із виникненням великих мовних моделей (англ. Large Language Model, LLM), які є дуже великими моделями глибокого навчання, які заздалегідь навчені на величезних обсягах даних [2, 3, 4]. Великі мовні моделі неймовірно гнучкі. Одна модель може виконувати різні завдання, такі як відповіді на питання, узагальнення документів, мовні переклади і складання пропозицій. Їхня здатність розуміти природну мову та генерувати зв’язний текст робить їх революційним досягненням у сфері штучного інтелекту [5, 6].

Ці моделі активно застосовуються в бізнесі, освіті, медицині, творчості, аналітиці, кібербезпеці та багатьох інших галузях. LLM можуть кардинально вплинути на створення контенту та використання людьми пошукових систем та віртуальних помічників для користувача ПК. Великі мовні моделі можуть робити це завдяки мільярдам параметрів, які дозволяють їм уловлювати складні закономірності мови та виконувати широкий спектр мовних завдань. LLM здійснюють революцію у додатках у різних галузях: від чат-ботів та віртуальних помічників до створення контенту, допомоги у дослідженнях та мовного перекладу. Продовжуючи розвиватися та вдосконалюватися, LLM готові змінити способи нашої взаємодії з технологіями та доступом до інформації, зробивши їх ключовою частиною сучасного цифрового ландшафту.

Трансформер, що лежить в основі LLM – це набір нейронних мереж, кожна з яких складається з кодера і декодера з можливістю самонавчання (рис. 1.1). Кодер і декодер отримують значення з послідовності тексту і розуміють співвідношення між словами і фразами.

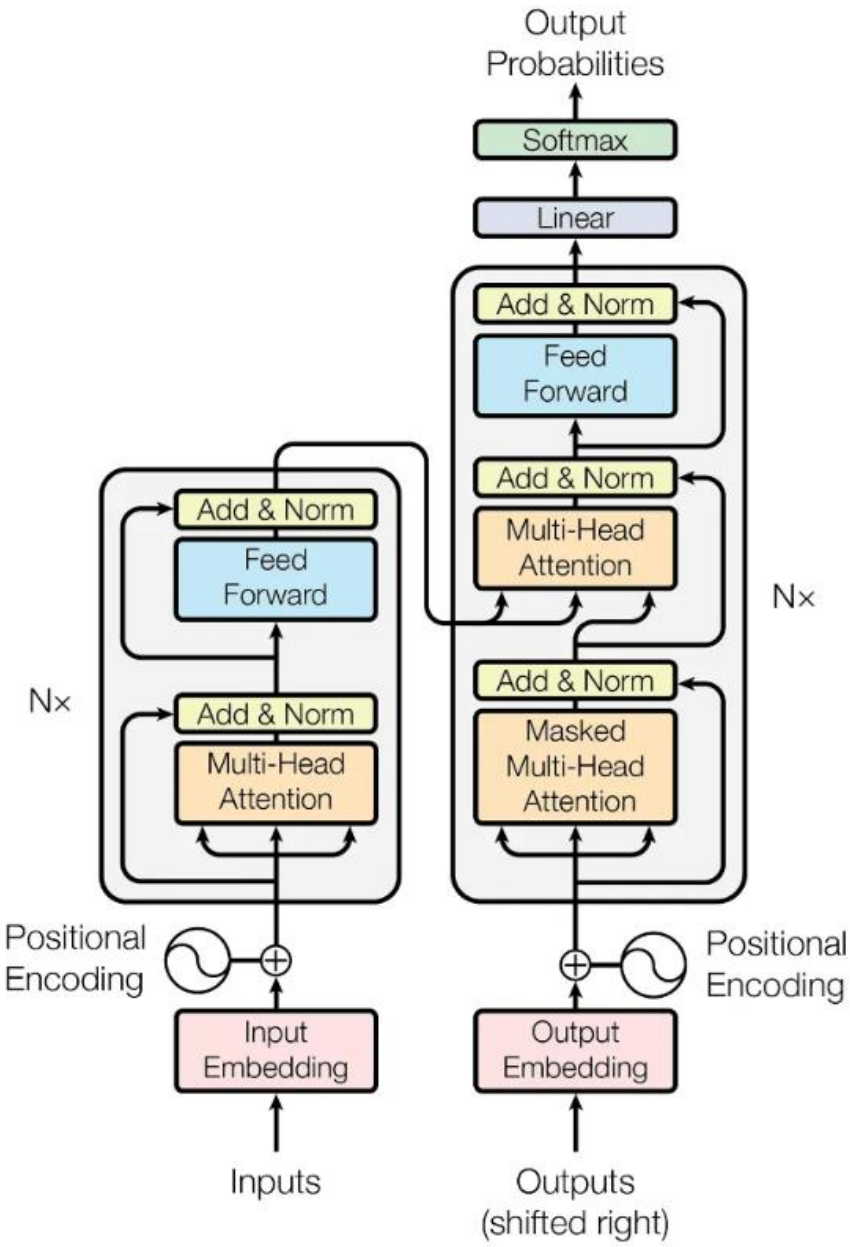


Рисунок 1.1 – Модель архітектури трансформера

Трансформери здатні навчатися без навчання, точніше, трансформери здійснюють самонавчання [7]. Саме завдяки цьому процесу трансформери вчаться розуміти базову граматику та мови, а також засвоювати знання. На відміну від

попередніх рекурентних RNN-мереж, які послідовно обробляють вхідні дані, трансформери обробляють цілі послідовності паралельно. Це дозволяє фахівцям обробки даних використовувати графічні процесори для навчання моделей на основі трансформерів, що значно скорочує час навчання. Архітектура нейронної мережі трансформера дозволяє використовувати дуже великі моделі, часто із сотнями мільярдів параметрів. Такі надвеликі моделі можуть отримувати величезні обсяги даних, часто з Інтернету, а також з таких джерел, як індекс Common Crawl, що налічує понад 50 мільярдів веб-сторінок, та Вікіпедія, що налічує близько 57 мільйонів сторінок.

Впровадження трансформерів стало одним із ключових проривів, що привело до створення моделей GPT (англ. Generative Pre-trained Transformer). На рисунку 1.2 зображено еволюція моделей на основі архітекрути Transformer.

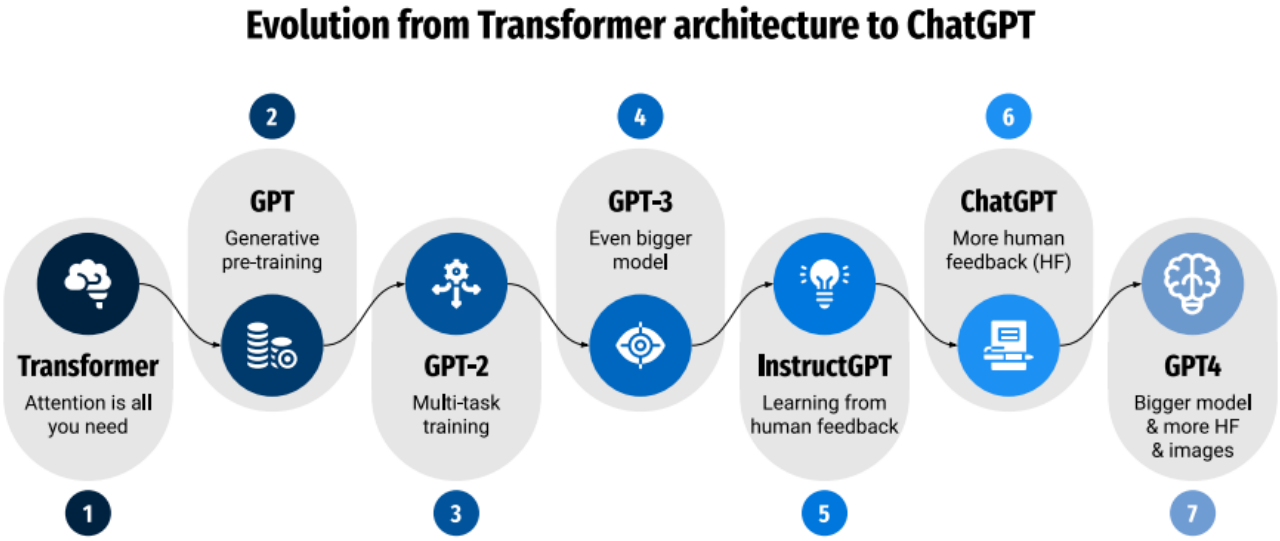


Рисунок 1.2 – Еволюція моделей на основі архітектури Transformer

В 2019 році OpenAI презентувала мовну модель Generative Pre-Trained Transformer 2, або GPT-2. На відміну від існуючих трансформерів, ця нейромережа була у змозі створювати довгі рядки пов’язаного тексту, відповідати на запитання, створювати вірші та генерувати тексти відповідної тематики. У 2020 році OpenAI продемонструвала нову версію GPT-3. Модель GPT-3 Open AI має 175 млрд

параметрів. ChatGPT, може визначати закономірності на основі даних і генерувати природні та легкочитані вихідні дані. Хоча розмір Claude 2 не є відомим, він може приймати на вході до 100 тисяч токенів у кожному запиті і, відповідно, працювати із сотнями сторінок технічної документації чи навіть цілою книгою. Наступник, GPT-4, продовжив цю традицію, забезпечуючи ще кращу якість генерації тексту та можливість навчання на ще більшій кількості даних.

Провідні великі технологічні компанії стали демонструвати власні розробки в області великих мовних моделей. До найбільш популярних LLM належать GPT-3 і GPT-4 від OpenAI, BERT і PaLM від Google, а також LLAMA, створена Meta [8, 9, 10]. IBM запустила серію моделей Granite, яка стала основою генеративного штучного інтелекту для інших продуктів IBM, таких як watsonx Assistant та watsonx Orchestrate. Модель Jurassic-1 від AI21 Labs має 178 мільярдів параметрів, словниковий запас із 250 000 слів та аналогічні розмовні можливості. Модель Cohere Command має аналогічні можливості і може працювати більш ніж 100 різними мовами. Компанія LightOn Paradigm пропонує базові моделі із заявленими можливостями, що перевершують можливості GPT-3. Всі ці LLM поставляються з API, які дозволяють розробникам створювати унікальні програми для генеративного штучного інтелекту [11, 12].

Таким чином, здійснений огляд нейромережевих моделей для обробки природної мови дозволив установити, що застосування передових технологій обробки природної мови – нейромережевих моделей на основі трансформерів, дозволяє створити віртуальний помічник, здатний виконувати аудіокоманди операційної системи з високою точністю, що робить їх ідеальними для системи підтримки користувача ПК у операційній системі Windows.

1.2 Велика мовна модель LlaMA

Модель LlaMA (англ. Large Language Model Meta AI) є власною розробкою компанії Meta AI, яка належить Meta (раніше Facebook) і займається розробками у

сфері штучного інтелекту, а також технологій доповненої та штучної реальності. Відмінністю моделі LlaMA є відкритість, що дозволяє розробникам досить вільно її завантажувати та використовувати. Що є контрастом з такими моделями, як Claude від Anthropic, GPT-4 від OpenAI (двигун ChatGPT), а також Gemini від Google, які доступні виключно через API [9, 10, 12 13].

Модель LlaMA випускається з 2023 року й є модель NLP з мільярдами параметрів, навчена 20 мовах, допомагає створювати чат-ботів швидше та економніше, ніж раніше (рис. 1.3).

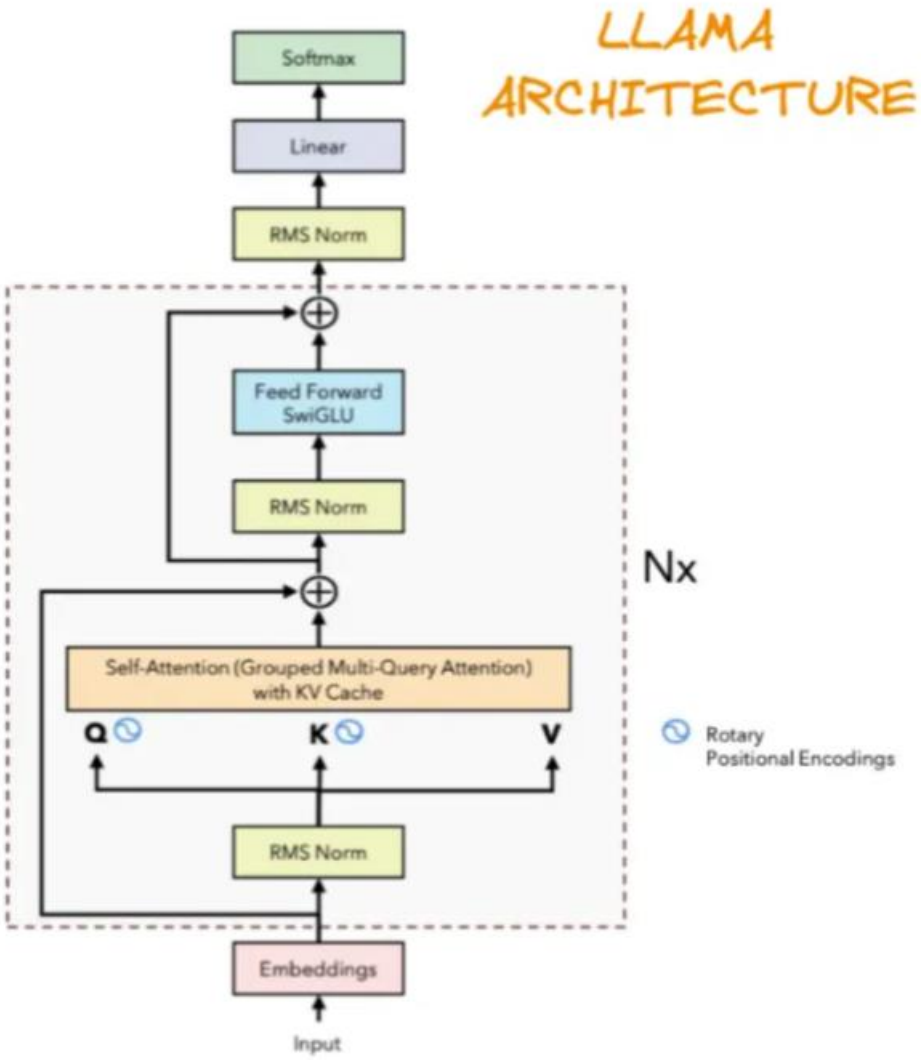


Рисунок 1.3 – Архітектура моделі LlaMA

Для надання розробникам більше гнучкості, Meta уклала партнерські відносини з постачальниками хмарних послуг, таких як AWS, Google Cloud та Microsoft Azure, для пропозиції хмарних версій LlaMA. До того ж Meta створила інструменти, які спрощують розробникам процес точного налаштування та адаптації моделі у відповідності з їх конкретними вимогами. До особливостей моделі відносять [14]:

– *оптимізовану продуктивність*: LlaMA спроектована таким чином, щоб працювати швидше та ефективніше навіть на обмежених обчислювальних ресурсах;

– *розширювану архітектуру*: її легко адаптувати до нових завдань і інтегрувати в різні інфраструктури, що робить її універсальною для застосувань;

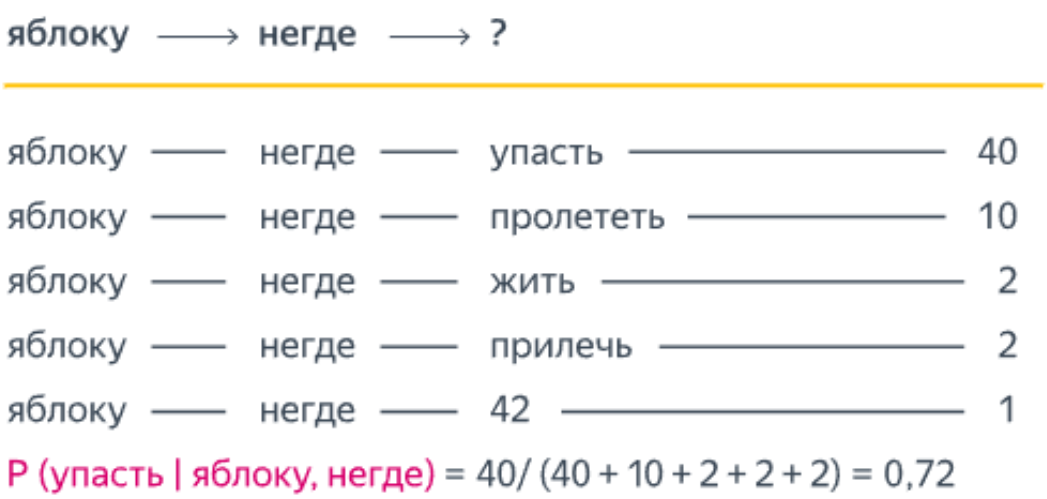
– *багатомовність*: завдяки своїм параметрам LlaMA може ефективно працювати з різними мовами, адаптуючись до потреб користувачів.

Самі останні версії LlaMA (LlaMA 3.1 8B, LlaMA 3.1 70B, LlaMA 3.1 405B) були випущені влітку 2024 року. Ці моделі навчаються на широкому спектрі джерел даних, таких як вебсторінки на різних мовах, загальнодоступний код, доступні в Інтернет файли і синтетичні дані, створені іншими системами ШІ.

LlaMA, як і інші моделі трансформерів, використовує механізм уваги, який дозволяє фокусуватися на ключових елементах тексту. Великі мовні моделі, як і усі мовні моделі, використовують поняття токена. Під час токенізації текст розбивається на найменші смислові одиниці тексту, які розуміють алгоритми – токени. Кожен токен перетворюється у числове представлення, яке моделі використовують для обчислень. Завдяки цьому LlaMA може працювати з будь-якими мовами, навіть із тими, на яких вона не навчалася безпосередньо, адаптуючи свій алгоритм до нових даних через самонавчання [10, 15].

До найбільш поширених алгоритмів токенізації підслів, що використовуються в моделях Transformer, відносять Unigram, WordPiece, BPE та SentencePiece.

Раніше у статистичних мовних моделях речення розбивалося на слова за пробілами, розділовими знаками, виключалися стоп-слова і так далі. Цей підхід є досить популярним досі, наприклад, у розумних клавіатурах для підказки наступного слова. Такий алгоритм токенізації має назву Unigram (рис. 1.4).



(*токен – окреме слово*)

Рисунок 1.4 – Приклад Unigram-токенізації

Використання Unigram-токенізації базується на таких перевагах: простоті імплементації, високій швидкості роботи алгоритму, низькій обчислювальній вартості навчання та інтерфейсу. Однак цей метод: не може генерувати слова, які не ідуть підряд у навчаючому наборі даних;

– має дуже малий контекст;

– ймовірність дуже довгих послідовностей близько до нуля, тому алгоритм не може видавати розумні речення довгої довжини.

До головних недоліків такого підходу відносять дві проблеми з різними словоформами, які:

– або позначаються різними токенами, що не зовсім правильно, адже слово одне і те ж, тому виходить, що відразу кілька токенів мають схожий зміст;

– або приводяться до початкової форми, що призводить до втрати відмінку, часу, числа.

Більш сучасні токенізатори побудовані на алгоритмі BPE (англ. Byte Pair Encoding). Рішення вимагає фіксації певного числа токенів. Як тільки це зроблено, до словника додаються всі символи з тексту, шукаються найчастіші поєднання і знову додаються. Цей процес триває до того часу, поки кількість токенів стане однаково заданому значенню. На рисунку 1.5 показано різницю підходів до токенізації у різних алгоритмах.

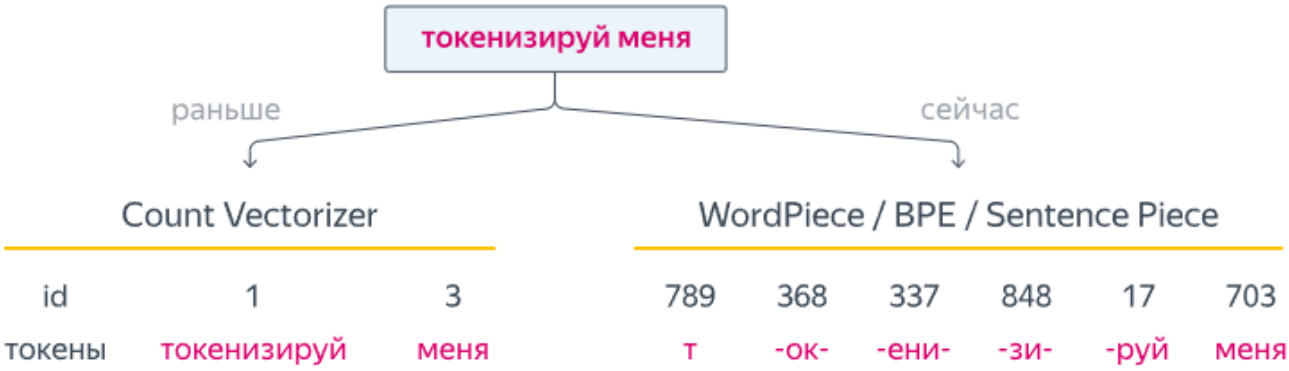


Рисунок 1.5 – Порівняння моделей токенізації токенізації

Токенізатор SentencePiece є досконалішим за BPE – він, успадковуючи логіку токенізаторів Unigram і BPE, по іншому працює з пробілами (додає «_» перед відповідним токеном) і не побудований на логіці розбиття слів по роздільникам. Тому, на відміну від BPE, він здатний працювати навіть з такими мовами, як японська чи китайська, де символ пробілу не використовується. Іншою головною особливістю SentencePiece є обернена токенізація: оскільки в ньому немає спеціальної обробки пробілів, декодування токенів здійснюється просто шляхом їхньої конкатенації та заміни _ на пробіли.

Усі моделі LlaMA мають контекстне вікно на 125 000 токенів, що дозволяє їм обробляти близько 100 000 слів (300 сторінок тексту) [9]. Довге контекстне вікно допомагає моделі зберігати інформацію з останніх документів і даних, знижуючи

риск відхилення від теми. LlaMA 3.1 8B і 70B – компактні моделі, які можуть працювати з різними пристроями від ноутбуків до серверів. LlaMA 3.1 405B є крупномасштабною моделлю, для якої зазвичай необхідним є обсладнання ценру обробки даних. Однак усі моделі забезпечують більш високу продуктивність та оптимізовані для зменшення обсягу пам’яті і затримки.

LLaMa може бути реалізована у інтелектуальних системах допомоги для виконання контекстних та точних дій у відповідь на аудіозапити користувачів. Це може бути корисно в обслуговуванні користувачів ПК, де потрібна симуляція людської взаємодії з підтримкою приодної мови.

1.2 Існуючі програмні рішення з підтримкою голосового спілкування

Система, яка є помічником, що підтримує аудіо спілкування з користувачем природною українською мовою та реалізує голосове управління ПК в операційній системі Windows передбачає створення віртуального асистента, який дозволяє управляти ПК за рахунок голосових команд на українській мові.

Голосові асистенти є невід’ємною частиною майже всіх великих IT компаній – це реалії сьогодення. Проєкти та технології компаній-гігантів Apple, Google, Microsoft та Amazon вражають. Кожен день є повідомлення про вдосконалення старих продуктів та створення нових, які за своїм функціоналом випереджають час.

Більшість віртуальних помічників є безкоштовними, оскільки вони вже інтегровані в пристрої користувачів. Тому, для того, щоб користуватись помічником, необхідним є пристрій, який вже має його, або на який його можна встановити. Тут хорошим прикладом є Siri, яка стала невід'ємним елементом iOS. Розумний помічник доступний на більшості пристроїв iPhone, iPod, iPad, Apple Watch.

Google також не відстає від своїх конкурентів і його власна продукція є не менш потужною. Google Assistant доступний майже на всіх смартфонах під операційною системою Android, а також на ПК. Проте, щоб увімкнути його на телефоні потрібно змінити мову системи на ту, яку він підтримує. На жаль, серед

таких немає української. Але компанія не здається і активно працює над тим, щоб найближчим часом українці змогли відчути всі переваги їхнього продукту на собі.

Однак є і платні голосові асистенти. Одними із найдорожчих виступають помічники для керування пристроями розумного будинку. Розумна колонка Google Home дозволяє користувачам використовувати голосові команди, які реалізовує Google Assistant. Зовнішні та внутрішні сервіси компанії, інтегровані в систему, дозволяють користувачам відтворювати відео чи фото, слухати музику чи просто дізнаватись новини. Google Home підтримує функції автоматизації дому, що робить його одним із найсильніших конкурентів на світовому ринку. Перший пристрій був випущений ще у 2016 році, а початкова ціна становила 129$. Зараз ціна може сягати і 500$. Звичайно такий помічник може стати в пригоді, але це за умови, що є кошти на його покупку.

Технології розвиваються із шаленою швидкістю, простим користувачам важко за всім встигати, а що ж говорити про людей із обмеженими можливостями. Навіть якщо вони розуміють принцип функціонування того чи іншого пристрою і знають як ним користуватись, то що робити, коли їхні фізичні можливості обмежені. Саме тут на допомогу приходять спеціальні пристрої і голосові асистенти не стали винятком. Завдяки їм люди можуть легко керувати смартфонами, комп’ютерами та пристроями розумного будинку.

Siri (англ. Speech Interpretation and Recognition Interface) – хмарний персональний помічник, що також є системою взаємодії з користувачем. Цей застосунок використовує розпізнавання природної мови, щоб давати рекомендації і відповіді на питання. Siri індивідуально пристосовується до кожного користувача, вивчаючи його риси протягом довгого часу. Спочатку Siri повинна була бути доступною на телефонах BlackBerry та Android, проте потім ці плани було скасовано. Адже в квітні 2010 року корпорація Apple купила Siri Incorporation, відтоді Siri стала доступною лише на продуктах компанії Apple. На даний момент віртуальний помічник доступна на таких платформах: iPhone, iPad, iPod touch, Macintosh, Apple TV, Apple Watch, HomePod. Написана на мові програмування

Objective-C. А розробниками сучасної версії асистента є власне компанія Apple, SRI International, Adam Cheyer, Dag Kittlaus.

Зараз Siri є невід'ємною частиною операційної системи IOS. Вона доступна на iPad третього покоління та вище, на версіях, пізніших iPhone 4S, iPod touch п'ятого покоління та iPad mini всіх поколінь. З листопада 2011 року компанія офіційно заявила, що не буде інтегрувати Siri у старіші версії iPhone, адже у них відсутній чип фільтрації фонового шуму. Голосовий асистент підтримує такі мови: англійська, французька, німецька, японська, іспанська, італійська, корейська, китайська, російська. Можливості Siri:

− може зателефонувати на потрібний вам номер або відправити за вас повідомлення;

− прокладати маршрут, показує, як дістатися до місця і час у дорозі;

− створює будильник: помічнику повідомлять, в який час потрібно прокинутися – він заведе будильник, можна просто поставити часовий проміжок «розбуди мене через годину»;

− планує календар, що необхідно діловим і забудькуватим людям, фіксує час заходу і нагадує про нього власнику iPhone;

− Apple Music і Siri відмінно працюють разом;

− є можливість швидко перевіряти факти, робити розрахунки, переводити фрази з однієї мови на іншу, достатньо лише попросити;

− навіть коли немає питань, Siri виконує обов'язки особистого помічника.

Голосовий помічник Siri є душе ефективним та корисним у користуванні. Проте великим недоліком є те, що він не є доступним користувачам Android та Windows. А також не підтримує багатьох мов світу.

Google Assistant – хмарний сервіс персонального асистента, розроблений компанією Google, був представлений у травня 2016 року на презентації Google I/O. Він вважається продовженням більш раннього Google Now, але в ньому з'явилася можливість участі у двосторонній розмові. Помічник може використовуватися в смартфонах, також він включений в Google Allo – застосунок для миттєвого обміну

повідомленнями, Google Home – розумний голосовий Wi-Fi динамік для управління будинком, Android Wear – розумний годинник від Google.

Google Assistant підключається до Google Now і може отримувати з нього інформацію, виводячи її в привабливішому вигляді для користувача, перевіряти погоду і багато чого ще. Однак, на відміну від своїх аналогів, він може брати участь у розмові із користувачем, використовуючи алгоритм обробки природної мови Google. Продовження розмови без повтору фрази «OK, Google» на даний момент є лише англійською мовою. З листопада 2017 Google Assistant може розпізнавати пісні, які грають поруч. Досить сказати «Що це за пісня?» або «Яка пісня грає?». Google Assistant був запущений з використанням голосу Кікі Бесселл для американського жіночого голосу, тієї ж актриси для системи голосової пошти Google Voice з 2010 року.

В жовтні 2019 року Google оголосив, що Icca Рей була додана в Google Assistant в якості додаткового голосового помічника, який міг бути включений користувачем, сказавши «Окей, Google, говори як Issa». Починаючи з серпня 2018 року асистент почав працювати і на території України, але без підтримки української мови. Однак зараз вже доступна, і з кожним запитом стає все кращою. На жаль, повна версія є недоступною в Україні. Адже одна й та ж функція працює по-різному в різних країнах. Проте є багато різноманітних корисних функцій, які може виконувати помічник, від звичайних «подзвони комусь/відкрий застосунок» до більш складних «Створи список покупок і додай туди масло, яйця та молоко». Ви також можете попросити його увімкнути відео на Youtube, знайти пісню, провести зарядку для очей чи запитати яка сьогодні погода. У складі Google Allo асистент прослуховує, зберігає та систематизує всі діалоги користувачів з метою пошуку ключових слів і понять. Для деяких повідомлень він може автоматично пропонувати різні варіанти отримання послуг чи товарів.

Google Assistant написаний намові програмування C++ та підтримується наступними операційними системами: Android, IOS, Wear OS, Android TV, Android Auto, Chrome OS. Недоліком віртуального помічника є те, що його досить важко

встановити на ПК. А для того, щоб це зробити, необхідні відповідні знання в галузі IT. Також компанія Google може стежити за користувачами при використанні Google Home, і відмовилась від додавання спеціальної кнопки, яка дозволить відключити мікрофон за бажанням користувача.

Amazon Alexa, або Alex – віртуальний помічник, розроблений компанією Amazon, який вперше з'явився в розумних колонках Amazon Echo і Amazon Echo Dot. Асистент підтримує голосове спілкування, відтворення музики, підкастів та аудіокниг, складання списків справ, налаштування будильників, надання актуальної інформації про погоду, трафік, спорт, новини тощо, управління пристроями в розумному домі. Користувачі можуть розширювати можливості Alexa, встановлюючи навички, розроблені сторонніми постачальниками.

Більшість пристроїв з Alexa розпочинає діалог за допомогою активаційної фрази (наприклад, «Alexa»). Інші, наприклад, смартфон – вимагають натискання кнопки включення. Alexa доступна англійською, німецькою, французькою, італійською, іспанською та японською мовами. У січні 2019 року команда Amazon прозвітувала про продаж понад 100 мільйонів пристроїв з підтримкою Alexa.

Microsoft Cortana – це віртуальний асистент із впровадженими елементами штучного інтелекту. Вона була представлена на конференції Build в квітні 2014 року в Сан-Франциско. Зараз доступна на Windows 10, Android, Microsoft Band, Windows Phone 8.1. Програма названа в честь ігрового персонажа серії відеоігор Halo відMicrosoft, озвученого актрисою Джен Тейлор. Cortana може передбачати потреби користувача. Для цього їй необхідно надати доступ до особистих даних, наприклад, історія пошуків у мережі, адреснакнига, електронна пошта. Цифрова помічниця інтегрована в «Пошук» у Windows 10 і не є окремою програмою. Вона активується при зверненні до пошуковика, тому команди їй можна давати як голосові, так і текстові. Пошук здійснюється за допомогою систем Bing, Foursquare і серед особистих файлів. Сама компанія Microsoft позиціонує Cortana як голосового помічника, який допоможе вам виконувати більше справ, витрачаючи при цьому менше часу. Наприклад, вона організовує роботу в Windows 10 або буде

керувати пристроями розумного будинку.

Ще одним прикладом голосового асистента є Аліса – віртуальний голосовий помічник, створений компанією Яндекс. Розпізнає природне мовлення, імітує живий діалог, дає відповіді на запитання користувача та завдяки запрограмованим навичкам вирішує прикладні та важкі завдання. Аліса працює на смартфонах, планшетах, комп'ютерах та автомобілях. За даними Яндекса, місячна аудиторія Аліси станом на грудень 2019 року склала 45 млн осіб за більш ніж мільярд запитів за рік.

Всі віртуальні помічники підтримують одні й ті ж базові функції, будь це налаштування будильника, відправлення електронних листів, відтворення музики тощо. Але у кожного асистента також є особливості, які роблять їх оригінальними.

Унікальність голосового асистента Cortana полягає у тому, що він:

− використовує власну пошукову систему Bing, у той час як Siri і Google Assistant використовують Google, Cortana для відповіді на запити застосовує Bing від Microsoft;

− інтегрований в усі пристрої Windows 10: всі ПК з Win10 і консоль Xbox One, Cortana також живе в декількох інтелектуальних пристроях, включаючи динамік Harmon Kardon Invoke;

− є більше, ніж голосовий асистент, тому що покладається не тільки на голосові команди, як наприклад Siri, а й на введені команди;

− знає про користувача більше, ніж він сам – у Cortana є унікальна функція «Нотатки», яку Microsoft додала до програмного забезпечення після спілкування з реальними особистими помічниками і може запам'ятовувати переваги та уподобання користувача;

− відстежує посилки та поїздки, може сканувати електронні листи для отримання інформації про рейси, автоматично тримає користувача в курсі планів його поїздок;

− Cortana завжди працює, постійно включена на відміну від Siri, яка починає працювати натисканням кнопки «Додому».

Ще одна особливість – обмеження у віці. Якщо в акаунті Microsoft вказати вік менше 13 років, то скористатися послугами помічника не можна.

Cortana є невід'ємною частиною ОС і не поширюється в якості окремого додатка. У 2015 році компанія представила «Microsoft Launcher» в «Google Play». Асистент виконує всі основні функції, але як і раніше не підтримує українську мову. Якщо завантажити Cortana на айфон, то вона не привнесе якихось особливих функцій, але елементи від Microsoft будуть забезпечені. Все ж, фірмовий асистент Siri набагато зручніший і корисніший на айфон.

Завантажити Cortana для iPhone і iPad можна з App Store, але тільки в деяких країнах: США, Канаді і деяких інших. Голосовий асистент оптимізується для певних пар «мова-регіон». Найкраще голосовий помічник працює, коли в смартфоні або комп'ютері мовні та регіональні налаштування збігаються. Саме тому, головним недоліком Cortana є те, що вона не підтримує багатьох мов світу, в тому числі і українську. Перелік мов, які вона підтримує: англійська, французька, німецька, італійська, іспанська, традиційна та спрощена китайська, португальська, японська.

Незважаючи на незаперечні переваги голосових помічників у них є і ряд недоліків, найбільшим з яких є неправильне розпізнавання голосових команд. Так, у 2019 році було проведено дослідження від аналітичної компанії Perficiend Digital, який із помічників можна назвати найбільш передовим. У тесті взяли участь Cortana, Google Assistant, Alexa та Siri. Сірим кольором позначені повні правильні відповіді, а червоним – спроба відповісти. Як видно з рисунку 1.6, Google Assistant випереджає інших. Він відповів правильно майже на 90% запитань, а це свідчить не лише про коректність роботи самого асистента, а і про грамотність та вміння чітко розпізнавати людський голос.

У версії ОС Windows 11 Microsoft випустила оновлення, яке робить голосовий помічник Cortana непрацездатним. Компанія офіційно припиняє підтримку віртуального асистента. В операційній системі Windows 10, яка отримуватиме оновлення до жовтня 2025 року, голосовий асистент Cortana, як і

раніше, працює. А в ОС Windows 11 його підтримка уже призупинена. Замість Cortana в Windows 11 з'явився Windows Copilot, який використовує алгоритми штучного інтелекту та відповідає на запитання користувача у стилі ChatGPT.

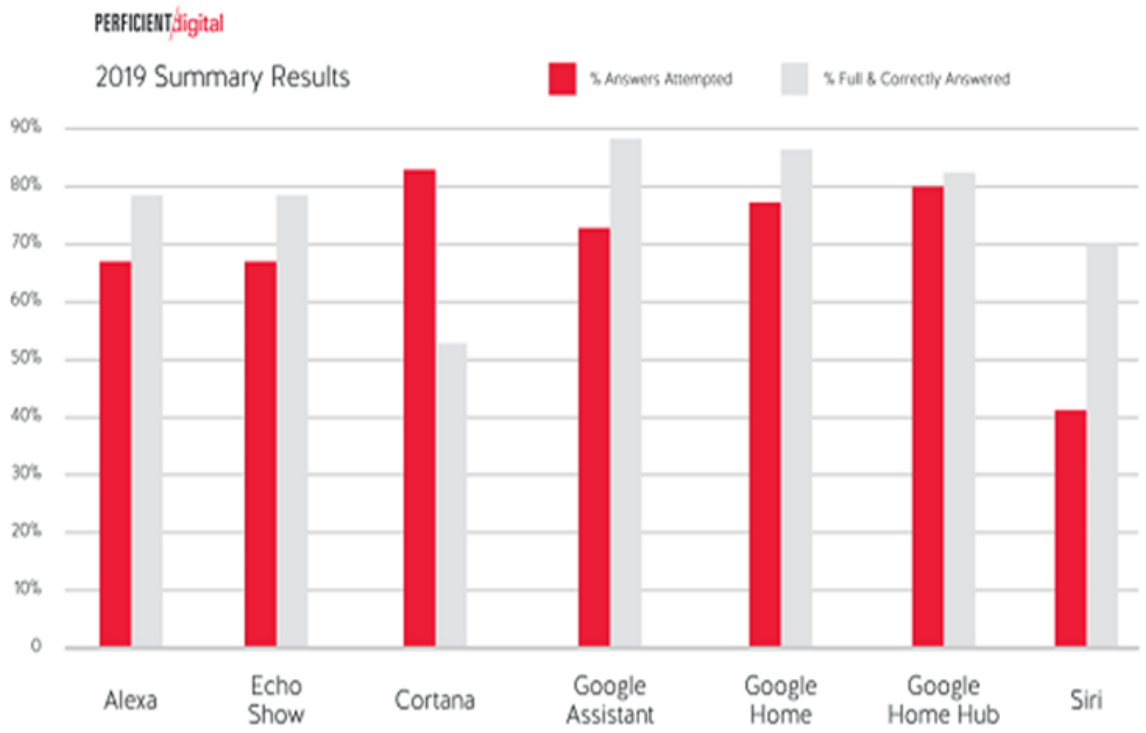


Рисунок 1.6 – Дослідження голосових помічників від Perficiend Digital

Кнопка виклику Copilot в інтерфейсі Windows 11 розташовується на панелі задач. Помічник здатний відповідати на запитання користувача, шукати різну інформацію в інтернеті. Крім того він може виконувати Windows-специфічні команди: на запит перемикати тему оформлення інтерфейсу ОС з темною на світлу і навпаки, робити знімки екрана, а також включати режим «Не турбувати».

Голосовий асистент Windows 11 дозволяє виконувати багато команд користувальницького інтерфейсу по управлінню ПК тільки за рахунок голосу: розпізнає голос, вимикає ПК, відкриває та закриває програми, створює й перейменовує файли і папки, відкриває сторінки у браузері, набирає текст та багато іншого. Однак все це працює тільки на англійській мові, підтримки української мови немає. Copilot Microsoft можна буде використовувати для створення

електронних листів, нагадувань, пошуку в Мережі та інших завдань. Підтримка багатьох цих функцій поки що не реалізована в новому Windows Copilot, хоча в перспективі він обіцяє стати значно ефективнішим у порівнянні з Cortana.

Здійснений аналіз голосових помічників користувачів дозволив установити, що голосова підтримка більшою мірою реалізована у мобільних операційних системах. Їх функціональність для підтримки користувача операційної системи ПК є досить обмеженою. Це обумовлено складністю задачі голосового управління ПК. Серед іншого можна також відмітити високу ціну за якісний продукт, а для українських користувачів ще й недоступність спілкування з голосовими помічниками українською мовою.

1.3 Постановка задачі

Провівши аналіз підходів до підтримки аудіоспілкування користувача по виконанню команд користувальницького інтерфейсу в операційній системі Windows було зроблено висновок про необхідність розробки інтелектуальної системи супроводження користувача ПК із використанням методів обробки природної української мови на основі нейромережевих моделей.

Створення інтелектуальної системи полягає у розробці прототипу системи, що включатиме адаптивну нейромережеву модель для підтримки користувачів під час роботи з ПК. Передбачено також проведення оцінки ефективності системи в умовах реального використання для осіб із обмеженими можливостями.

Вимоги до системи – це властивості, якими повинна володіти система, щоб програмне та апаратне забезпечення могло безперебійно та ефективно працювати. Недотримання цих вимог може привести до проблем з установкою програмного забезпечення чи його продуктивністю. Під час реалізації даної системи потрібно приділити основну увагу розпізнаванню людського голосу та правильності його обробки.

Якщо на апаратному рівні мікрофон буде відсутній чи пошкоджений,

користувач не зможе використовувати всіх можливостей системи. З іншої сторони виступає програмний рівень. Він добре продуманий, адже існують не тільки вже готові бібліотеки для розпізнавання людської мови, а й додаткові модулі, які дозволяють приглушувати сторонній шум. Розробку вимог до голосового помічника можна розділити на декілька етапів: знаходження вимог (визначення потреб користувачів та систем); аналіз вимог (перевірка на адекватність); специфікація (документування); тестування вимог.

Для створення голосового асистента було встановлено наступні вимоги:

- можливість точного розпізнавання людського голосу на близькій відстані;
- приглушення стороннього шуму;
- розпізнавання голосовим асистентом української мови;
- швидке виконання команд користувача;
- безперебійна робота;
- можливість модернізації.

Очікуваним результатом є створення помічника для користувача ПК, який має розширену функціональність по управлінню ПК за допомогою голосових команд: вимикає ПК, відкриває та закриває програми, створює, перейменовує, переміщує, копіює або видаляє файли і папки, відкриває сторінки у браузері, набирає текст, контролює відтворення відео та аудіо, зупинку та пропуск треків.

Об’єктом дослідження є процес виконання команд користувальницького інтерфейсу в операційній системі ПК.

Предметом дослідження є програмні засоби та нейромережеві моделі для голосового спілкування з користувачем по виконання команд UI-інтерфейсу в операційній системі Windows.

Мета дослідження – підвищення ефективності взаємодії користувача з операційною системою ПК шляхом створення інтелектуальної системи із використанням методів обробки природної української мови на основі нейромережевих моделей.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

– дослідити теоретичні засади голосового спілкування з користувачем по виконанню команд UI-інтерфейсу ОС Windows із використанням нейромережевих моделей і методів обробки природної мови та здійснити аналіз існуючих рішень;

– обґрунтувати вибір технологій та засобів розробки системи підтримки користувача ПК;

– здійснити проєктування та програмну реалізацію інтелектуальної системи супроводження користувача ПК у вигляді голосового асистента з підтримкою української мови та оцінити її ефективність.

Висновки до розділу 1

Досліджено теоретичні засади підтримки UI-інтерфейса користувача в операційній системі Windows із використанням нейромережевих моделей обробки природної мови. Установлено, що революційні зміни в системах обробки природної мови обумовлені виникненням великих мовних моделей (LLM), архітектура яких базується на трансформерах – великих наборах нейронних мереж із можливістю самонавчання. Архітектура трансформерів дозволяє використовувати дуже великі нейронні моделі з сотнями мільярдів параметрів і підтримкою багатьох мов. Здійснений огляд нейромережевих моделей для обробки природної мови дозволив установити, що застосування передових технологій обробки природної мови – нейромережевих моделей на основі трансформерів, дозволяє створити віртуальний помічник, здатний виконувати аудіокоманди операційної системи з високою точністю, що робить їх ідеальними для системи підтримки користувача ПК у операційній системі Windows. Серед LLM-моделей було виділено модель LlaMA, яка є моделлю NLP з мільярдами параметрів, навчена 20 мовах, має оптимізовану продуктивність, реалізована з використанням більш досконалих алгоритмів

токенізації й може бути реалізована у інтелектуальних системах допомоги для виконання контекстних та точних дій у відповідь на аудіозапити користувачів.

У контексті інтелектуальних систем супроводження користувача нейромережі дозволяють: автоматично вивчати індивідуальний стиль роботи користувача та підлаштовувати інтерфейс під його потреби; прогнозувати можливі дії користувача та надавати йому відповідні підказки чи допомогу; забезпечувати безперервну адаптацію системи до змін у поведінці чи стані користувача, що особливо важливо для осіб, які мають обмежені фізичні та ментальні можливості. Це дозволяє значно спростити взаємодію з комп'ютером для людей, які мають труднощі у його використанні, та підвищує їхню продуктивність.

Здійснений аналіз існуючих програмних рішень голових асистентів (Siri, Google Assistant, Amazon Alexa, Microsoft Cortana) дозволив установити, що голосова підтримка більшою мірою реалізована у мобільних операційних системах. Їх функціональність для підтримки користувача операційної системи ПК є досить обмеженою. Більшість з них є безкоштовними, а вартість якісних продуктів є високою і для українських користувачів ще й недоступність спілкування з голосовими помічниками українською мовою. Тому є потреба у створення системи підтримки користувача ПК із використанням методів обробки природної української мови на основі нейромережевих моделей.

2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ СИСТЕМИ ПІДТРИМКИ КОРИСТУВАЧА ПК

2.1 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) – це потужний редактор коду, розроблений компанією Microsoft, який надає широкий спектр інструментів для розробників програмного забезпечення. Створений на основі Electron Framework, VS Code працює на основних операційних системах, таких як Windows, Linux і macOS. Його популярність зумовлена багатьма функціями, які полегшують процес програмування, підвищують продуктивність та забезпечують зручність використання. Основними функціями є:

– *підсвічування синтаксису*: VS Code підтримує синтаксис багатьох мов програмування, що робить код легким для читання та розуміння. Це підвищує ефективність роботи розробників, адже підсвічування допомагає швидше виявляти помилки.

– *інтелектуальне завершення коду*: завдяки вбудованим алгоритмам автозавершення, редактор може підказувати варіанти для змінних, функцій та інших елементів коду, що значно скорочує час написання коду та знижує ймовірність помилок;

– *налагодження*: VS Code забезпечує інтегровані інструменти для налагодження, що дозволяє розробникам ефективно виявляти та виправляти помилки. Користувачі можуть встановлювати точки зупину, переглядати значення змінних у реальному часі та виконувати код покроково;

– *рефакторинг коду*: редактор надає зручні інструменти для рефакторингу, які дозволяють змінювати структуру коду без ризику його порушення, це включає безпечне перейменування змінних, видалення непотрібних фрагментів коду та об'єднання функцій;

– *вбудований Git*: VS Code інтегрує систему контролю версій Git, що дає

можливість розробникам зручно управляти змінами в коді, створювати гілки та об'єднувати їх, а також виконувати інші операції безпосередньо в редакторі.

Налаштування та розширення VS Code надає користувачам можливість змінювати налаштування інтерфейсу, такі як тема, комбінації клавіш та параметри. Це забезпечує персоналізацію середовища розробки відповідно до індивідуальних уподобань. Крім того, редактор підтримує установку розширень, що додають нові функції та інтегрують різні інструменти. На ринку розширень доступна велика кількість плагінів, які можуть додавати нові мовні модулі, інструменти для налагодження, підтримку фреймворків, а також можливості для роботи з базами даних та хмарними сервісами.

Visual Studio Code отримав визнання у спільноті розробників завдяки своїм численним перевагам. За даними опитування Stack Overflow 2022 року, яке охоплювало 71 010 респондентів, 74,48% учасників відзначили, що активно користуються VS Code, що свідчить про його високу популярність і довіру серед професіоналів у сфері програмування.

Таким чином, Visual Studio Code є ефективним інструментом для розробки програмного забезпечення, який надає широкий спектр функцій, що задовольняють потреби сучасних розробників. Завдяки своїй універсальності, доступності та потужності, він стає важливим елементом у процесі розробки, особливо в контексті створення інтелектуальних систем, таких як голосові асистенти.

2.2 Мова програмування Python та бібліотеки для автоматизації роботи з графічним інтерфейсом

Python є високорівневою мовою програмування загального призначення, яка також активно використовується для розробки інтелектуальних систем, включаючи системи супроводження користувача ПК на основі нейромережевих моделей [4, 22]. Ця мова сприяє підвищенню продуктивності розробників та полегшує читабельність коду, що є критично важливим для розробки складних систем.

Python підтримує різноманітні парадигми програмування, такі як структурне, об'єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване програмування. Це забезпечує гнучкість у виборі підходів до розробки, що дозволяє адаптувати рішення до конкретних вимог системи [16, 17].

Головними переваги Python вважаються:

– *простота та читабельність коду*: Python має зрозумілий синтаксис, що робить його доступним для новачків і дозволяє розробникам швидко писати та підтримувати код.

– *велика бібліотека стандартних модулів*: Python постачається з багатим набором стандартних бібліотек, що дозволяє використовувати готові рішення для виконання різних завдань без необхідності створювати їх з нуля.

– *підтримка різних парадигм програмування*: Python дозволяє використовувати об'єктно-орієнтоване, імперативне, функціональне та інші стилі програмування, що забезпечує гнучкість у виборі підходу до розробки.

– *кросплатформеність*: Python працює на різних операційних системах (Windows, Linux, macOS), що робить його універсальним для розробки програмного забезпечення.

– *широка спільнота*: Велика спільнота користувачів забезпечує доступ до безлічі ресурсів, форумів і документації, що полегшує пошук допомоги та рішень.

– *інтеграція з іншими мовами*: Python може бути легко інтегровано з іншими мовами програмування, такими як C, C++ та Java, що дозволяє використовувати його в складних проектах.

– *можливості для машинного навчання та штучного інтелекту*: Python є однією з найпопулярніших мов для розробки систем машинного навчання та штучного інтелекту завдяки своїм потужним бібліотекам і фреймворкам.

– *широкий вибір інструментів для розробки*: Python підтримує безліч інтегрованих середовищ розробки (IDE) та редакторів коду, таких як PyCharm, Jupyter Notebook та Visual Studio Code, що забезпечує комфортну роботу

розробника.

Основні властивості Python, такі як динамічна типізація, автоматичне управління пам'яттю та можливість самоорганізації, значно спрощують процес написання коду. Крім того, Python має вбудовану обробку винятків, підтримку паралельних обчислень та практичні структури даних на високому рівні, що робить його ідеальним вибором для створення інтелектуальних систем, здатних обробляти великі обсяги даних та виконувати складні обчислення.

Було відібрано також ряд бібліотек Python, які надають допомогу у створенні помічника користувача ПК в операційній системі Windows: os, webbrowser, Tkinter, PyAutoGUI [18].

Бібліотека Python ***os*** є однією з найважливіших і найчастіше використовуваних бібліотек, оскільки надає функції взаємодії з операційною системою [19]. Вона надає безліч функцій, які дозволяють нам взаємодіяти з базовою операційною системою, включаючи доступ та керування файловою системою, змінними середовищами і навіть процесами.

Зручний контролер браузера ***webbrowser*** надає високорівневий інтерфейс, який дозволяє відображати вебдокументи для користувачів та переглядати вебсторінки.

Бібліотека ***Tkinter*** дозволяє створювати програми з графічним інтерфейсом. Перевагами бібліотеки вважаються вбудованість у стандартний пакет Python (нічого додатково встановлювати не потрібно) та кросплатформеність, що дозволяє писати програми для різних операційних систем. Tkinter дозволяє створити повноцінний працюючий інтерфейс користувача мобільного або десктопного застосунку за допомогою вже вбудованих у програмний пакет класів та об'єктів, що їх наслідують [18].

Бібліотека Python ***PyAutoGUI*** призначена для автоматизації роботи з графічним інтерфейсом операційної системи. Вона дозволяє програмно імітувати дії користувача, такі як переміщення курсору, клацання миші, натискання клавіш клавіатури, а також інші взаємодії з графічним інтерфейсом користувача GUI.

PyAutoGUI підтримує створення скриптів, які автоматично виконують різні завдання, що особливо корисно під час рутинних операцій або тестування програмного забезпечення. Завдяки своїй простоті та інтуїтивно зрозумілому API, PyAutoGUI стає відмінним інструментом для автоматизації робочих процесів на комп'ютері, сприяючи виконанню монотонних завдань і підвищуючи продуктивність [18].

PyAutoGUI надає можливість автоматизувати ряд дій на комп'ютері, що робить його відтвореним інструментом у різних об'єктах. Ось деякі з основних областей застосування PyAutoGUI:

– *автоматизація рутинних завдань*: PyAutoGUI дозволяє створювати скрипти для виконання рутинних завдань, таких як запуск програм, копіювання та вставка даних, виправлення повідомлень та багато іншого. Це допомагає заощадити час і зменшити ймовірність помилок при виконанні повторних операцій.

– *тестування програмного забезпечення*: PyAutoGUI підтримує автоматизацію тестування програмного забезпечення, включаючи GUI-тестування. Створення скриптів дозволяє проводити автоматичні тести на додатках, перевіряти їх функціональність і стабільність.

– *створення ботів і макросів*: за допомогою PyAutoGUI можна розробляти боти для виконання визначених завдань в автоматичному режимі. Наприклад, боти можуть автоматизувати процеси веб-скрапінгу, управління ігровими персонажами або виконання дій у додатках.

– *віддалене керування*: PyAutoGUI може використовуватися для віддаленого керування комп'ютером, що корисно в місцях, коли необхідно автоматизувати дії на віддаленій машині чи сервері.

У цілому PyAutoGUI є потужним інструментом для автоматизації роботи з графічним інтерфейсом, який може значно полегшити виконання різних завдань і підвищити ефективність досвіду користувача. PyAutoGUI пропонує широкий спектр можливостей для керування мишкою, клавіатурою та іншими елементами

графічного інтерфейсу користувача (GUI). Деякі з основних можливостей PyAutoGUI включають:

– *управління мишкою*: за допомогою PyAutoGUI можна переміщати курсор миші на екран, виконувати кліки та подвійні кліки, а також виконувати скролінг, це дозволяє автоматизувати дії, які зазвичай виконуються користувачем вручну;

– *управління клавіатурою*: Бібліотека дозволяє емулювати натискання клавіш клавіатури, введення тексту, а також використовувати комбінації клавіш, це корисно при створенні скриптів для автоматизації введення даних або керування додатками через клавіатуру;

– *робота з зображеннями*: PyAutoGUI надає можливість пошуку та відтворення зображень на екрані, це може бути використано для визначення розташування елементів на екрані, а також для виконання дій на основі виявлених зображень;

– *робота з вікнами*: бібліотека дозволяє отримувати інформацію про різні вікна та елементи GUI, такі як заголовки вікон, координати елементів та інші характеристики, це полегшує взаємодію з застосунками та оточуючими елементами на екрані.

Завдяки цим можливостям PyAutoGUI стає потужним засобом для автоматизації роботи з графічним інтерфейсом і надає широкі можливості для створення скриптів автоматизації та оптимізації робочих процесів на комп’ютері.

2.3 Бібліотеки для обробки природної мови

Для обробки голосових команд користувача було проаналізовано для використання наступні бібліотеки: Pocketsphinx**,** SpeechRecognition, Pyttsx3, Hugging Face, Transformers. Розглянемо їх більш детально.

Pocketsphinx – це потужна бібліотека для розпізнавання мови, яка є частиною проекту CMU Sphinx, розробленого в Університеті Карнегі-Меллона. Вона спеціально оптимізована для роботи на пристроях з обмеженими ресурсами, що

робить її ідеальним вибором для створення мобільних та настільних додатків, включаючи інтелектуальні системи супроводження користувача. Основні особливості Pocketsphinx:

– *легка вага*: Pocketsphinx має малий розмір та потребує менше системних ресурсів у порівнянні з багатьма іншими системами розпізнавання мови, що дозволяє використовувати її на менш потужних пристроях;

– *офлайн-робота*: бібліотека дозволяє розпізнавати мову без необхідності підключення до Інтернету, що робить її зручною для використання в умовах обмеженого або відсутнього доступу до мережі;

– *гнучкість*: Pocketsphinx підтримує різні мови та акценти, а також надає можливість навчання та налаштування моделей для специфічних вимог. Це дозволяє адаптувати систему до потреб користувачів;

– *простота інтеграції*: бібліотека легко інтегрується з Python та іншими мовами програмування, що спрощує процес розробки програмних рішень;

– *висока точність розпізнавання*: Pocketsphinx демонструє високу точність розпізнавання мови, що робить її надійним вибором для реалізації голосових команд у системах супроводження користувача;

– *велика спільнота та документація*: бібліотека має активну спільноту користувачів, що сприяє обміну досвідом та ідеями. Документація Pocketsphinx містить багато прикладів та інструкцій, що полегшує процес навчання та використання бібліотеки.

У контексті інтелектуальної системи супроводження користувача, Pocketsphinx доцільно використовувати для реалізації голосового управління. Це дозволяє користувачам взаємодіяти з системою за допомогою усних команд, що значно підвищує зручність та доступність. Наприклад, система може розпізнавати команди на кшталт "відкрити програму", "знайти документ" або "програти музику", що дозволяє користувачеві ефективно керувати ПК без необхідності використовувати клавіатуру або мишу.

SpeechRecognition – це бібліотека Python, що забезпечує простий інтерфейс для розпізнавання мови з різних джерел. Вона підтримує декілька популярних API для розпізнавання мови, включаючи Google Web Speech API, Microsoft Azure Speech API та інші [20]. Це робить бібліотеку універсальним інструментом для розробки голосових асистентів і систем, які взаємодіють з користувачами за допомогою голосових команд. Основні особливості SpeechRecognition:

– *підтримка різних джерел звуку*: бібліотека може обробляти звукові файли (наприклад, WAV, AIFF) або записувати звук безпосередньо з мікрофона, що забезпечує гнучкість у виборі джерела;

– *інтеграція з різними API*: SpeechRecognition підтримує кілька платформ для розпізнавання мови, таких як Google Cloud Speech API, IBM Watson, Microsoft Bing Voice Recognition та інші, що дозволяє легко переключатися між ними залежно від потреб проекту;

– *можливість роботи в офлайн-режимі*: хоча більшість API потребують доступу до Інтернету, бібліотека також підтримує деякі локальні рішення для розпізнавання, що дозволяє розпізнавати мову без необхідності постійного підключення до мережі;

– *простота використання*: інтерфейс бібліотеки є інтуїтивно зрозумілим, що дозволяє швидко впроваджувати розпізнавання мови у свої програми без великих зусиль;

– *висока точність*: SpeechRecognition використовує передові технології для забезпечення високої точності розпізнавання мови, зокрема сучасні алгоритми обробки сигналів і машинного навчання;

– *підтримка багатьох мов*: бібліотека підтримує широкий спектр мов, що робить її універсальним вибором для міжнародних проектів.

У рамках інтелектуальної системи супроводження користувача бібліотека SpeechRecognition використана для розпізнавання голосових команд, які дозволяють користувачам взаємодіяти з системою. Наприклад, користувач може сказати: «відкрий браузер», «пошук в Google» або «відтворити музику», а система,

використовуючи цю бібліотеку, може точно розпізнати ці команди та виконати відповідні дії.

Pyttsx3 – це Python-бібліотека для синтезу мови, яка дозволяє перетворювати текст у звучання. На відміну від багатьох інших бібліотек для синтезу мови, pyttsx3 працює офлайн, що робить її зручною для використання у додатках, де немає доступу до Інтернету. Вона підтримує кілька мов і голосів, що дозволяє розробникам налаштовувати звучання для своїх програм. Основні особливості pyttsx3:

– *офлайн-робота*: pyttsx3 не потребує підключення до Інтернету, оскільки синтез мови виконується локально на пристрої, це забезпечує більшу гнучкість та швидкість, оскільки немає затримок через мережеві запити.

– *підтримка різних платформ*: бібліотека працює на різних операційних системах, включаючи Windows, macOS та Linux. Це дозволяє використовувати її в кросплатформених програмах.

– *налаштування голосу***:** pyttsx3 дозволяє розробникам змінювати параметри голосу, такі як швидкість, гучність та вибір конкретного голосу з доступних на системі. Це дозволяє налаштовувати звучання відповідно до потреб проекту.

– *підтримка декількох мов*: Бібліотека може працювати з різними мовами, що дозволяє створювати програми, здатні озвучувати текст різними мовами, що робить її універсальним інструментом для розробки мультимовних додатків.

– *простота використання***:** Інтерфейс pyttsx3 є інтуїтивно зрозумілим і простим, що дозволяє легко інтегрувати синтез мови у власні програми без великих зусиль.

У рамках інтелектуальної системи супроводження користувача бібліотека pyttsx3 може бути використана для озвучування повідомлень, підказок або відповідей на запити користувача. Наприклад, система може повідомити користувачу про виконання команди або надати інструкції, що підвищує зручність та доступність інтерфейсу.

Бібліотека ***Hugging Face*** застосовується для роботи з моделями NLP на Python [21]. У своїх розробках Hugging Face використовує технологію ШІ та сервіси розпізнавання мови і створення тексту. Hugging Face прагне створити моделі обробки природної мови NLP доступними для всіх і надає ряд ресурсів із відкритим вихідним кодом, щоб користувачі могли розробляти моделі та розміщувати їх у проектах за доступними цінами. Продукты Hugging Face спрощують процес машинного навчання та обробки природного мови. Для цього вони пропонують:

– велику кількість попередньо навчених моделей;

– інструменти для точних налаштувань цих моделей під вимоги кожного проекту;

– зручні варіанти використання моделей в різних середовищах.

Бібліотека ***Transformers***, підтримувана Hugging Face, є потужним інструментом для різних програм в галузі обробки природної мови, комп'ютерного зору та обробки звуку [7]. Платформа Hugging Face – це колекція готових сучасних попередньо навчених Deep Learning моделей LLM. А бібліотека Transformers надає інструменти та інтерфейси для їх простого завантаження та використання. Це дозволяє економити час і ресурси, необхідні для навчання моделей з нуля [22].

Transformers не є набором модулів, з яких складається нейронна мережа, наприклад PyTorch. Натомість Transformers надає кілька високорівневих абстракцій, які дозволяють працювати з моделями в кілька рядків коду. Моделі вирішують дуже різноманітний спектр завдань:

– NLP: класифікація, відповідь на запитання, мовне моделювання, конспектування, переклад, множинний вибір, генерація тексту;

– класифікація,виявлення об'єктів, сегментація;

– audio: класифікація, розпізнавання мови;

– відповіді на запитання в таблиці, оптичне розпізнавання символів, витяг інформації зі сканованих документів, відеокласифікація, візуальні відповіді на питання.

Одне й те завдання може вирішуватися різними архітектурами великих мовних моделей, список яких нині є більшим за 150. Найбільш відомі: Vision Transformer (ViT), T5, ResNet, BERT, GPT2. На цих архітектурах навчено понад 60000 моделей. Моделі Transformers підтримують три фреймворки: PyTorch, TensorFlow і JAX. Для PyTorch'а доступні майже всі архітектури. Також моделі можна експортувати у формати ONNX та TorchScript.

2.4 Технологія Google Cloud Speech API

Google Cloud Speech API є хмарним сервісом, розробленим компанією Google, який дозволяє програмам і системам здійснювати автоматичне розпізнавання мови в реальному часі. Ця технологія використовує сучасні алгоритми машинного навчання та нейронні мережі, що робить її однією з найефективніших у своєму класі. Основні компоненти технології:

– *висока точність розпізнавання*: завдяки використанню нейронних мереж та алгоритмів машинного навчання, API забезпечує високу точність розпізнавання мови, що робить його ідеальним для застосувань, де необхідно точно інтерпретувати усні команди або транскрибувати розмови;

– *підтримка багатьох мов*: Google Cloud Speech підтримує понад 125 мов і діалектів, що дозволяє використовувати API в різних країнах і для різних користувачів;

– *гнучке налаштування*: розробники можуть адаптувати API для роботи з конкретними термінами або фразами, що покращує якість розпізнавання в специфічних галузях, таких як медицина, техніка чи юриспруденція;

– *аудіоформати*: API підтримує різноманітні формати аудіофайлів, такі як WAV, FLAC, MP3, що робить його зручним для використання в різних контекстах;

– *розпізнавання в реальному часі*: Google Cloud Speech може обробляти аудіопотоки в реальному часі, що відкриває можливості для створення

інтерактивних голосових асистентів та інших систем, які потребують швидкої реакції на усні команди;

– *API для інтеграції*: розробники можуть легко інтегрувати Google Cloud Speech у свої програми за допомогою RESTful API або бібліотек для популярних мов програмування.

Перевагами Google Cloud Speech API є:

– *висока точність*: використання передових алгоритмів машинного навчання забезпечує високу точність розпізнавання мови.

– *гнучкість*: API може бути адаптоване для різних сценаріїв використання, включаючи розпізнавання мовлення в реальному часі та обробку попередньо записаних аудіофайлів.

– *підтримка множинних мов*: це робить його ідеальним для міжнародних проектів, де потрібна підтримка різних мов.

– *зручність у використанні*: простий інтерфейс API та велика кількість доступних бібліотек дозволяють розробникам швидко впроваджувати його в свої проект

2.5 Фреймворк Llamaindex

Llamaindex – це фреймворк, який надає зручні інструменти для роботи з великими мовними моделями. Він дозволяє індексувати великы обсяги текстових даних і виконувати по ним ефективний пошук. Це особливо корисно при роботі з LLM-моделями, які можуть генерувати текст на основі наданого контексту та обробляти природну мову [23].

Основна ціль Llamaindex – полегшити роботу з мовними моделями, надавши зручні інструменти для індексування та пошуку. Це дозволяє розробникам брати участь у створенні застсоунків і сервісів, що використовують мовні моделі, замість того, щоб тратити час і ресурси на вирішення технічних проблем.

Llamaindex заснований на концепції Retrieval Augmented Generation (RAG). В основі RAG лежить ідея про те, що для генерації відповіді на запит можна використовувати не тільки контекстний запит, але й додаткові дані, знайдені за індексом (рис. 2.1). Це дозволяє генерувати більш точні та інформативні відповіді.

Рисунок 2.1 – Схема роботи Llamaindex

В рамках Llamaindex основні елементи RAG представлені наступними модулями:

– *з'єднувачі даних*: це модулі, які відповідають для завантаження та обробки даних, вони дозволяють завантажити дані, створити з них узли та побудувати індекс.

– *Retrievers*: це модулі, які відповідають за пошук за індексом, вони дозволяють отримати пошук за індексом і найбільш релевантні узли.

– *механізми запитів*: це модулі, які відповідають для генерації відповіді на запит, вони використовують інформацію, отриману від ретриверів, для генерації відповіді.

В Llamaindex query_engine – це об’єкт, який керує процесом пошуку в індексі. Індекс являє собою структуру даних, яка зберігає «вузли» – Nodes (рис. 2.2). Вузол відповідає фрагменту тексту з документа. LlamaIndex приймає об’єкти документів і внутрішньо розбиває їх на об’єкти вузлів. Коли викликається query_engine.query('ваш запит'), задається питання ('ваш запит'), і query_engine шукає найбільш релевантні вузли в індексі, які можуть відповісти на поставлене запитання. Це робиться шляхом складання запиту з кожним вузлом в індексі та вибору найбільш релевантних узлів.

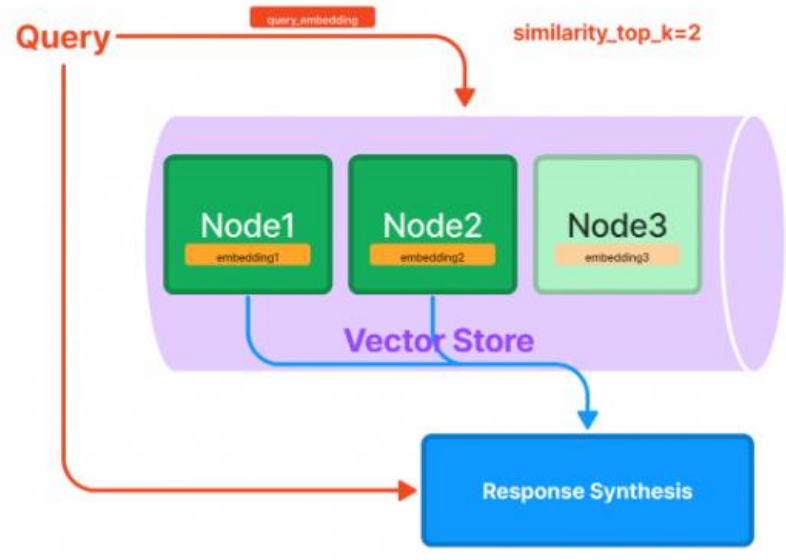


Рисунок 2.2 Робота query_engine

Важливо відзначити, що релевантність тут визначає модель, яка навчається приймати, які вузли можуть бути релевантними для певного запиту. Це може включити в себе аналіз семантичного змісту запиту та вузлів, а також інші фактори.

У результаті query_engine.query('ваш запит') повертає відповідь на запит, який створено на основі найбільш релевантних вузлів.

У LlamaIndex є різні типи індексів, кожен з яких працює трохи по-різному. Наприклад, індекс використання сховища (VectorStoreIndex) зберігає кожен вузол і відповідне йому векторне зображення, а при пошуку повертає топ-k найбільш схожих узлів. Другий тип індексу, індекс таблиці ключових слів (KeywordTableIndex), витягує ключові слова з кожного вузла та будує відображення кожного ключового слова до відповідного узла цього ключового слова. При пошуку він витягає релевантні ключові слова із запиту і ищет відповідні узли.

Генерація відповіді. Після вилучення релевантної ноди дані потрапляють у Response Synthesis. Цей процес бере дані з вибраних узлів і використовує мовну модель для створення остаточної відповіді. Це може включати в себе об’єднання інформації з різних узлів та переформулювання інформації, щоб вона була більш зрозумілою тощо. Процес Response Synthesis може змінюватися в залежності від конкретної конфігурації:

– *default*: у цьому режимі для кожного узла робиться виклик окремої мовної моделі, це дозволяє створити і уточнити відповідь, послідовно проходячи через кожен узел. Цей режим хороший для більш детальних відповідей;

– *compact*: у цьому режимі під час кожного виклику мовної моделі «упаковується» максимальна кількість текстових блоків узлов, які можуть поміститися в межах максимального розміру запиту; якщо блоків занадто багато, щоб помістити їх в один запит, відповідь створюється та уточнюється, проходячи через кілька запитів;

– *tree_summarize*: у цьому режимі, враховуючи набір узлов і запит, рекурсивно будується дерево і в якості відповіді повертається корневий узел.

Ці режими можуть бути поєднані з іншими параметрами запиту, такими як required_keywords і exclude_keywords, які дозволяють відфільтрувати вузли на основі наявності або відсутності певних ключових слів.

Висновки до розділу 2

Для створення інтелектуальної системи підтримки користувача ПК було обрано мову програмування Python, середовище розробки Visual Studio Code. Для автоматизації роботи з графічним інтерфейсом операційної системи Windows обрано Python-бібліотеки os, webbrowser, Tkinter, PyAutoGUI. Бібліотека os надає безліч функцій, які дозволяють нам взаємодіяти з базовою операційною системою, включаючи доступ та керування файловою системою, змінними середовищами і процесами. Webbrowser надає високорівневий інтерфейс для відображення та перегляду вебсторінок. Tkinter дозволяє створити повноцінний працюючий інтерфейс користувача мобільного або десктопного застосунку. Бібліотека PyAutoGUI має багато засобів для автоматизації основних операцій з графічним інтерфейсом операційної системи – управління мишею, клавіатурою, роботи з вікнами та зображеннями.

Для обробки голосових команд користувача було використано бібліотеки Pocketsphinx**,** SpeechRecognition, Pyttsx3, Hugging Face, Transformers. Pocketsphinx застосовано для реалізації голосового управління. SpeechRecognition використана для розпізнавання голосових команд, які дозволяють взаємодіяти з системою. Бібліотека pyttsx3 дозволяє озвучувати повідомлення, підказки або відповіді на запити користувача. Transformers, підтримувана Hugging Face, використана для доступу до колекції навчаних нейромережевих моделей оброки природної мови.

Хмарний сервіс Google Cloud Speech API обрано для автоматичного розпізнавання мови в реальному часі. Для полегшення роботи з мовною моделлю LlaMA вирішено використовувати фреймворк Llamaindex.

3 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ СУПРОВОДЖЕННЯ КОРИСТУВАЧА ПК

3.1 Моделювання та проєктування системи

На етапі моделювання було сформульовано функціональні вимоги до системи. Сьогодні існує дуже мало віртуальних помічників, які можуть розуміти команди UI-інтерфейсу операційної системи Windows на українській мові, і ще менше тих, які можуть вести двосторонню розмову із користувачем. Тому була поставлена задача створення голосового асистента, який зможе розуміти українську мову.

Голосовий асистент повинен буде виконувати наступні функції:

– розпізнавати голосові команди користувача;
– надсилати листи на електронну пошту;
– шукати необхідну інформацію в мережі Інтернет;
– керувати живленням ПК;
– запускати різні програми;
– нагадувати користувачу про важливі події.

На етапі проєктування для вебзастосунку інтелектуальної системи розроблено IDEF0-діаграму, на якій показано загальну схему діяльності системи. Розробка власного голосового асистента передбачає розпізнавання голосових команд користувача. Після чого здійснюється їх обробка, сюди входить перевірка на коректність вхідних даних та звірення з базою даних команд. І в результаті виконання команди. Основні вимоги подані на діаграмі IDEF0 рисунку 3.1.

Діаграма потоків даних (англ. Data Flow Diagram) – це графічне представлення «потоків» даних в інформаційній системі. Вона також може використовуватися для візуалізації процесів обробки даних. На рисунку 3.2 зображено DFD-діаграму, на якій показано взаємодію системи із користувачем.

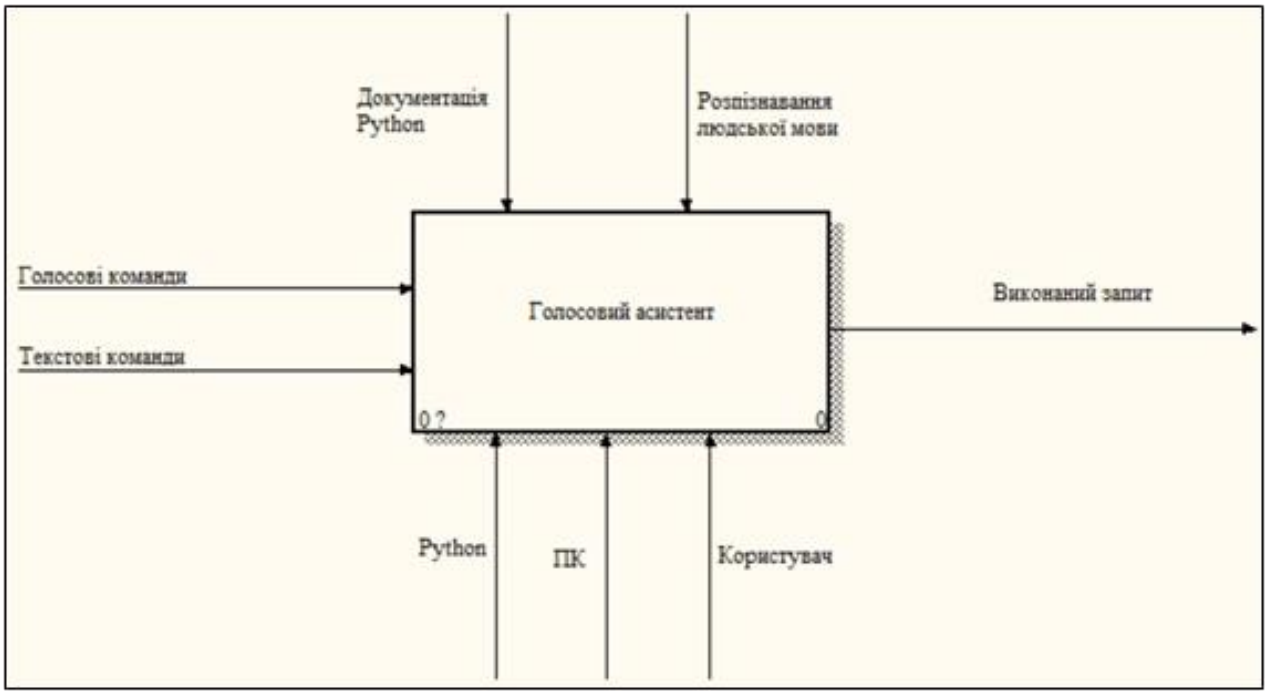


Рисунок 3.1 – IDEF0- діаграма

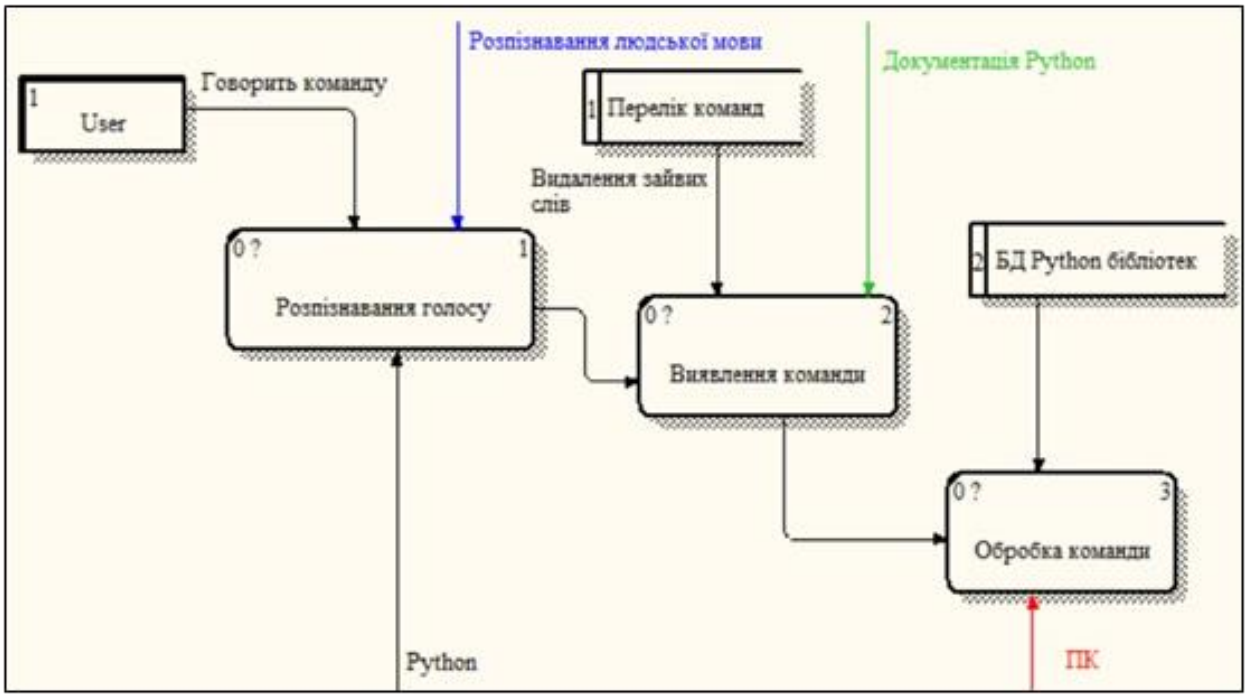


Рисунок 3.2 – DFD-діаграма взаємодії системи із користувачем

Проєктування системи включає побудову її архітектури, визначення основних компонентів та їх функцій. Архітектура системи супроводження

користувача ПК базується на модульному підході, що забезпечує гнучкість, масштабованість та легкість обслуговування. *Основні компоненти* системи включають:

— *модуль обробки природної мови* (NLP): здійснює обробку текстових команд та голосові запити користувача, аналізує вхідні дані та генерує відповіді;

— *модуль розпізнавання мови*: перетворює голосовий сигнал користувача на українській мові на текст і генерує є текстові команди для подальшої обробки;

— *модуль генерації мовлення*: озвучує текстові відповіді для користувача, що особливо зручно для людей з обмеженими фізичними можливостями;

— *модуль інтеграції з зовнішніми джерелами*: відповідає за отримання даних із зовнішніх API та баз даних;

— *модуль виконання команд*: інтегрується з операційною системою для виконання завдань користувача ПК.

Описана вище архітектура забезпечує поділ функціональності між різними компонентами системи та надає можливість у подальшому додавати нові функції або модифікувати існуючі.

3.2 Донавчання та оцінка якості моделі обробки природної мови

Голосовий асистент реалізовано з використанням великої мовної моделі LlaMA 3.2, яка є сучасним інструментом обробки природної мови (NLP) [21, 24]. Модель відповідає за аналіз введеного тексту, розпізнавання намірів користувача та генерацію відповідей на запити. Основні етапи роботи моделі LlaMA (табл. 3.1): 1) розпізнавання тексту, 2) попередня обробка, 3) аналіз тексту, 4) формування відповіді, 5) передача результату.

Основна функція моделі LLaMA – передбачати слово. На вхід дається послідовність слів, і моделі LLaMA видає наступне, найімовірніше слово, виходячи з тих текстів, на яких вона навчалася. Однак для створення гоосового асистента користувача ПК потрібно не отримання продовження послідовності слів, а

розпізнавання запитів користувача, які у модулі інтеграції з ОС будуть використані для виконання поставлених ним команд.

Для цього потрібне донавчання моделі. У даній роботі для створення голосового асистента, який спілкується з користувачем на українській мові, було використано уже навчену модель, яка може розуміти українську мову, взятій з платформи Hugging Face (https://huggingface.co/meta-llama/Llama-3.2-3B) [21]. Початкова версія моделі мала широкий набір знань, однак її функціонал було адаптовано для розуміння специфіки основних голосових команд UI-інтефейсу користувачів ПК шляхом донавчання.

Таблиця 3.1 – Основні етапи роботи моделі LlaMA

Етап	Опис
Розпізнавання тексту	Отримання текстового запиту, який був сформований модулем розпізнавання голосу
Попередня обробка	Очищення тексту від зайвих символів, токенізація, лемматизація
Аналіз тексту	Передача запиту до моделі Llama 3.2 для визначення намірів користувача та контексту
Формування відповіді	Генерація тексту, що відповідає на запит або описує наступну дію системи
Передача результату	Повернення згенерованого тексту іншим модулям (модулю виконання команд або модулю озвучення)

Донавчання моделі LLaMA здійснювалося з використанням LlamaIndex – фреймворку, призначеного для полегшення використання надбудов в моделі LLM, із використанням архітектури генерації відповіді, доповненої результатами пошуку (англ. Retrieval Augmentation Generation, RAG). RAG-архітектура поєднує LLM із зовнішніми базами знань і дозволяє додати в модель актуальну інформацію або якісь конкретні дані, на яких модель не вчилася спочатку (рис. 3.3). Це зручно при

роботі з інформацією, яка включає дані, специфічні для конкретних областей, що сприяє ефективному виконанню запитів [25].

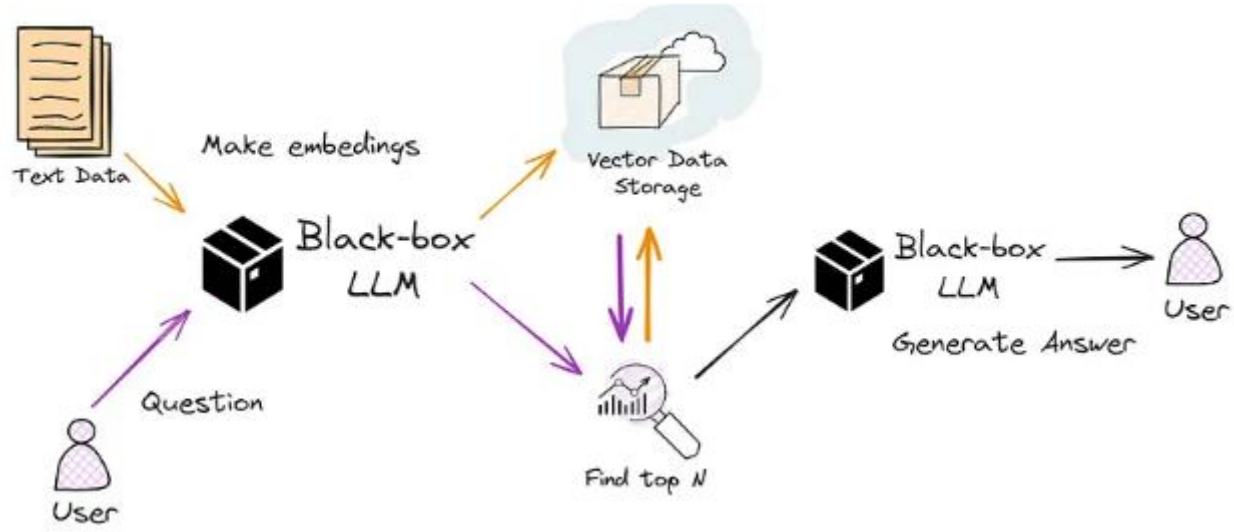


Рисунок 3.3 – Схема роботи RAG

Для адаптації моделі до задач голосового помічника користувача ПК у даному проєкті процес донавчання мовної моделі було здійснено на створеному DataSet текстового формату JSON, який охоплює типові команди, що надходять від користувачів ПК. На відміну від навчання базової моделі, донавчання набагато менш затратне.

Для підготовки DataSet було виконано збір та підготовка відповідних даних стосовно UI-інтерфейсу користувача ПК. Дані були розподілені за категоріями, що відображають основні функції помічника:

- OpenApplication – відкриття програм;
- SearchFiles – пошук файлів;
- ControlSettings – зміна налаштувань системи;
- PlayMedia – запуск медіафайлів;
- GetHelp – отримання довідки;
- Cancel – скасування дії;

– None – команда не розпізнана.

На етапі попередньої обробки текстів для підвищення якості навчання тексти команд були очищені від зайвих символів, приведені до нижнього регістру, а також пройшли токенізацію та лематизацію.

На етапі підготовки до навчання було підібрано оптимальні параметри, включаючи кількість епох, розмір батчу та швидкість навчання, щоб досягти балансу між продуктивністю та якістю роботи моделі.

Донавчання моделі відбувалося у кілька етапів. Опишемо їх детальніше.

1. *Імпорт даних*. Дані у форматі JSON було завантажено через API, що дозволяє використовувати Hugging Face Transformers для обробки текстових даних (рис 3.4). Кожен запис включає текст команди та відповідну категорію (напрямок дії).

```
import json
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments

# Завантаження даних
with open('commands_dataset.json', 'r') as f:
    dataset = json.load(f)

tokenizer = AutoTokenizer.from_pretrained('Llama-3.2')
model = AutoModelForSequenceClassification.from_pretrained('Llama-3.2', num_labels=7)

# Токенізація
def preprocess_data(data):
    return tokenizer(data['text'], truncation=True, padding=True, max_length=128)

tokenized_dataset = dataset.map(preprocess_data)
```

Рисунок 3.4 - Імпорт даних у навчальне середовище

2. *Навчання на адаптованому датасеті*. З цією метою використовувався метод fine-tuning(), який дозволив спеціалізувати модель під розпізнавання голосових команд (рис 3.5) [6].

3. *Тестування і валідація*. Було використано тестовий набір даних для перевірки точності розпізнавання команд та відповідності категорій.

```
training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset['train'],
    eval_dataset=tokenized_dataset['test']
)

trainer.train()
```

Рисунок 3.5 – Метод fine-tuning для навчання на адаптованому датасеті

Якість роботи донавченої мовної моделі оцінювалася за такими метриками:

- Recall – частка правильно розпізнаних команд від загальної кількості;
- Precision – точність визначення категорій команд;
- F1-score – збалансована метрика, що враховує Recall та Precision.

Результати оцінки якості донавченої моделі є наступними:

- Recall: 97.8%
- Precision: 98.3%
- F1-score: 98.0%

Ці показники демонструють здатність моделі ефективно розпізнавати голосові команди та інтерпретувати їхній зміст. Після завершення донавчання модель була інтегрована у систему голосового помічника для користувачів ПК.

3.3 Програмна реалізація

Процес розробки Python-застосунку починається з завантаження необхідних для розробки бібліотек. (рис.3.6).

```
import speech_recognition as sr
import subprocess
import webbrowser
import os
from gtts import gTTS
from transformers import AutoTokenizer, AutoModelForCausalLM
import tkinter
import requests
from pycaw.pycaw import AudioUtilities
import datetime
```

Рисунок 3.6 – Підключення бібліотек

Основними інструментами, що використовуються для створення інтелектуальної системи супроводження користувача ПК на основі нейромережевих моделей, є: Transformers, PyTorch, gTTS (Google Text-to-Speech), tkinter, webbrowser, os, speech_recognition, requests, pycaw.

Transformers: забезпечує інтерфейс для роботи з попередньо навченими моделями, такими як Llama 3.2. Ця бібліотека спрощує процес завантаження моделей, обробки тексту та генерації результатів [7].

PyTorch: використовується як базова бібліотека для створення нейромереж. Дозволяє виконувати обчислення на GPU та працювати з багатошаровими моделями машинного навчання.

GTTS (англ. Google Text-to-Speech): забезпечує функцію перетворення тексту в мовлення. Підтримує синтез українською мовою та створення аудіофайлів.

Tkinter: стандартна бібліотека Python для створення графічних інтерфейсів користувача. Використовується для створення віконного інтерфейсу десктопного застосунку.

Webbrowser: надає інструменти для відкриття веб-сайтів через стандартний браузер за запитом користувача [26].

Os: дозволяє виконувати операції з файлами та запускати інші програми в операційній системі [19].

Speech_recognition: бібліотека для розпізнавання мовлення. Використовується для перетворення голосових команд у текст [20].

Requests: використовується для виконання HTTP-запитів. Дає змогу інтегрувати API та отримувати дані з веб-сервісів.

Pycaw (англ. Python Core Audio Windows): забезпечує управління аудіо на рівні операційної системи. Використовується для регулювання гучності та інших аудіо параметрів.

Далі здійснюється підключення моделі для подальшої роботи з нею (рис. 3.7). На рисунку 3.7 видно що model_name вказує, яку модель використовуємо, AutoModelForCausalLM.from_pretrained (model_name) завантажує модель Meta-Llama-3-8B-Instruct, яка вже навчена для обробки мовлення і тексту. Це дозволяє підключити модель і використовувати її для генерації тексту на основі вхідного контексту.

```
# Модель та токенізатор для Meta-Llama-3-8B-Instruct
model_name = "meta-llama/Meta-Llama-3-8B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
```

Рисунок 3.7 – Підключення моделі Meta-Llama-3-8B

Після визначення функції для розпізнавання голосу, яка використовує бібліотеку speech_recognition, ця функція дозволяє перетворювати голосові команди в текст. На цьому етапі ми використовуємо її для отримання команди від користувача (рис 3.8). Ця функція виконує:

– *ініціалізацію розпізнавача голосу*: створює об'єкт Recognizer для роботи з мікрофоном та аудіопотоком;

– *збір аудіопотоку*: використовує мікрофон для збору аудіопотоку, який надалі буде оброблено;

– *розпізнавання голосу*: використовує API від Google для розпізнавання голосу українською мовою;

– *обробку помилок*: sr.UnknownValueError – якщо голос не вдалося розпізнати, sr.RequestError – якщо виникла помилка з підключенням до API;

– *повернення*: повертає розпізнаний текст або порожній рядок у разі помилки.

```python
# Функція для розпізнавання голосу
def listen():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Слухаю вашу команду...")
        audio = recognizer.listen(source)

    try:
        query = recognizer.recognize_google(audio, language="uk-UA")
        print(f"Ви сказали: {query}")
        return query
    except sr.UnknownValueError:
        print("Вибачте, я не зрозумів вашу промову.")
        return None
    except sr.RequestError:
        print("Вибачте, сталася помилка з сервісом розпізнавання мови.")
        return None
```

Рисунок 3.8 – Функція для розпізнавання голосу

Далі було додано функцію для отримання інформації про погоду з використанням Weather API (рис 3.9). Спочатку створюється функція def get_weather(city), яка надсилає запит до Weather API, використовуючи URL, щоб отримати актуальну інформацію про погоду для зазначеного місця. Ця функція повертає текстову відповідь з прогнозом.

Наступним кроком було додано функцію для регулювання гучності на комп’ютері (рис. 3.10).

```python
def get_weather(city):
    try:
        url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER_API_KEY}&units=metric&lang=uk"
        response = requests.get(url)
        data = response.json()

        if response.status_code == 200:
            temperature = data["main"]["temp"]
            description = data["weather"][0]["description"]
            humidity = data["main"]["humidity"]
            weather_info = (f"У місті {city.capitalize()} зараз {description}. "
                            f"Температура: {temperature}°C, вологість: {humidity}%.")
            return weather_info
        elif data.get("message"):
            return f"Помилка: {data['message']}"
        else:
            return "Не вдалося отримати інформацію про погоду."
    except Exception as e:
        return f"Сталася помилка: {e}"
```

Рисунок 3.9 – Функція для отримання інформації про погоду

```python
# Функція для регулювання гучності
def adjust_volume(amount):
    try:
        sessions = AudioUtilities.GetAllSessions()
        for session in sessions:
            volume = session.SimpleAudioVolume
            current_volume = volume.GetMasterVolume()
            new_volume = max(0.0, min(1.0, current_volume + amount))
            volume.SetMasterVolume(new_volume, None)
        speak(f"Гучність змінено на {int(new_volume * 100)} відсотків")
    except Exception as e:
        speak(f"Не вдалося змінити гучність: {e}")
```

Рисунок 3.10 – Функція для зміни гучності на комп’ютері

Також було додано функції для закриття всіх або тільки активного вікна (рис.3.11).

На Рисунку 3.12 показана функція для роботи з браузером та пошуку інформації в ньому.

Також було додано відкриття новин, мапи та переклад (рис. 3.13).

```python
# Функція для закриття активного вікна
def close_window():
    try:
        active_window = gw.getActiveWindow()
        if active_window:
            active_window.close()
            speak("Активне вікно закрито")
        else:
            speak("Немає активного вікна")
    except Exception as e:
        speak(f"Не вдалося закрити вікно: {e}")

# Функція для згортання всіх вікон
def minimize_all_windows():
    try:
        ctypes.windll.user32.ShowWindow(ctypes.windll.user32.GetForegroundWindow(), 6)
        speak("Усі вікна згорнуто")
    except Exception as e:
        speak(f"Не вдалося згорнути всі вікна: {e}")
```

Рисунок 3.11 – Функція для закриття вікон

```python
def process_command(command):
    if "відкрий google" in command or "відкрий будь ласка google" or "відкрий мені google" in command or "відкрий сам google" in command:
        speak("Відкриваю Google")
        webbrowser.open("https://www.google.com")
    elif "відкрий ютуб" in command or "відкрий будь ласка ютуб" or "відкрий мені ютуб" or "відкрий сам ютуб" or "відкрий youtube" in command or "відкрий бу
        speak("Відкриваю YouTube")
        webbrowser.open("https://www.youtube.com")
    elif "знайди на ютубі" in command or "знайди будь ласка на ютубі" or "знайди мені на ютубі" or "знайди сам на ютубі" or "знайди на youtube" or "знайди
        query = command.replace("знайди на ютубі", "").replace("знайди будь ласка на ютубі", "").replace("знайди на youtube", "").replace("знайди будь ласка
        if query:
            speak(f"Шукаю на YouTube: {query}")
            webbrowser.open(f"https://www.youtube.com/results?search_query={query}")
        else:
            speak("Що саме ви хочете знайти на YouTube?")
    elif "знайди в гуглі" in command or "знайди будь ласка в гуглі" or "знайди мені в гуглі" or "знайди сам в гуглі" or "знайди в google" or "знайди будь ла
        query = command.replace("знайди в гуглі", "").replace("знайди будь ласка в гуглі", "").replace("знайди в google", "").replace("знайди будь ласка в
        if query:
            speak(f"Шукаю в Google: {query}")
            webbrowser.open(f"https://www.google.com/search?q={query}")
        else:
            speak("Що саме ви хочете знайти в Google?")
```

Рисунок 3.12 – Функція для роботи та пошуку у браузері

```python
    elif "відкрий новини" in command or "відкрий будь ласка новини" or "відкрий мені новини" or "відкрий сам новини" in command:
        speak("Відкриваю новини")
        webbrowser.open("https://news.google.com")
    elif "відкрий мапу" in command or "відкрий карти" or "відкрий будь ласка мапу" or "відкрий будь ласка карти" or "відкрий мені мапу" in command or "відкр
        speak("Відкриваю карту")
        webbrowser.open("https://www.google.com/maps")
    elif "переклади" in command or "переклади будь ласка" or "переклади зараз" or "переклади мені" in command or "переклади сам" in command:
        query = command.replace("переклади", "").replace("переклади будь ласка", "").replace("переклади зараз", "").replace("переклади мені", "").replace("п
        if query:
            speak(f"Перекладаю: {query}")
            webbrowser.open(f"https://translate.google.com/?sl=auto&tl=uk&text={query}")
        else:
            speak("Що саме ви хочете перекласти?")
    else:
        speak("Команда не розпізнана. Можна спробувати ще раз.")
```

Рисунок 3.13 – Функція для відкриття новин, мапи та переклад

Наступним кроком було додано функцію для відкриття застосунку на комп'ютері користувача (рис. 3.14). На рисунку можна помітити функція робить пошук файлів за назвою, що відповідають команді так якщо знайдено кілька обирає перший.

```
# Функція для відкриття застосунку
def open_application(app_name):
    # Папки, у яких шукати застосунок
    search_paths = [
        os.path.expanduser("~/Desktop"),  # Робочий стіл
        "C:\\Program Files",  # Програмні файли
        "C:\\Program Files (x86)",  # Програмні файли (x86)
        "C:\\Users\\%USERNAME%\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs"  # Меню "Пуск"
    ]
    found_apps = []
    app_name_lower = app_name.lower()
    # Пошук файлів з назвою, що відповідає команді
    for path in search_paths:
        for root, dirs, files in os.walk(path):
            for file in files:
                if app_name_lower in file.lower() and file.endswith((".exe", ".lnk")):
                    found_apps.append(os.path.join(root, file))
    if found_apps:
        # Якщо знайдено кілька, беремо перший
        app_to_open = found_apps[0]
        try:
            subprocess.Popen(app_to_open, shell=True)
            speak(f"Відкриваю {app_name}")
        except Exception as e:
            speak(f"Не вдалося відкрити {app_name}: {e}")
    else:
        speak(f"Не вдалося знайти застосунок з назвою {app_name}. Перевірте, чи він встановлений.")
```

Рисунок 3.14 – Функція для відкриття застосунків

Було створено функцію для відкриття диску на комп'ютері та для відкриття папки на диску (рис. 3.15).

На рисунку 3.16 показана обробка для відкриття дисків та папок.

```python
# Функція для відкриття диска
def open_disk(disk_name):
    try:
        # Використовуємо команду для відкриття диску через провідник
        subprocess.run(f"explorer {disk_name}:", check=True)
        speak(f"Відкриваю диск {disk_name}")
    except Exception as e:
        speak(f"Не вдалося відкрити диск {disk_name}: {e}")

# Функція для відкриття папки на диску
def open_folder_on_disk(disk_name, folder_name):
    try:
        subprocess.run(f"explorer {disk_name}:{folder_name}", check=True)
        speak(f"Відкриваю папку {folder_name} на диску {disk_name}")
    except Exception as e:
        speak(f"Не вдалося відкрити папку {folder_name} на диску {disk_name}: {e}")
```

Рисунок 3.15 – Функція для відкриття дисків та папок

```python
# Обробка запитів для відкриття дисків
if "відкрий диск" in command:
    disk_name = command.replace("відкрий диск", "").strip()
    if disk_name:
        if disk_name in ['c', 'd', 'e', 'f']:  # Перевіряємо чи введена буква це валідний диск
            open_disk(disk_name)
        else:
            speak("Будь ласка, скажіть правильну букву диска, наприклад, C, D, E або F.")
    else:
        speak("Будь ласка, скажіть букву диска, який ви хочете відкрити.")
elif "відкрий папку" in command:
    parts = command.split("на диску", 1)
    if len(parts) == 2:
        disk_folder_part = parts[1].strip().split(" ", 1)
        if len(disk_folder_part) == 2:
            disk_name, folder_name = disk_folder_part
            if disk_name in ['c', 'd', 'e', 'f']:
                open_folder_on_disk(disk_name, folder_name.strip())
            else:
                speak("Будь ласка, скажіть правильну букву диска, наприклад, C, D, E або F.")
        else:
            speak("Будь ласка, скажіть правильну назву папки, яку ви хочете відкрити.")
    else:
        speak("Будь ласка, скажіть правильну команду, наприклад: 'відкрий папку на диску C' або 'відкрий папку на диску D'.")
else:
    speak("Команда не розпізнана. Можна спробувати ще раз.")
```

Рисунок 3.16 – Обробка для відкриття дисків та папок

Також було створено функцію для вимикання та перезавантаження комп’ютера (рис. 3.17).

```
# Функція для вимкнення комп'ютера
def shutdown_computer():
    try:
        # Для Windows
        subprocess.run(["shutdown", "/s", "/t", "1"], check=True)
        speak("Комп'ютер вимикається.")
    except Exception as e:
        speak(f"Не вдалося вимкнути комп'ютер: {e}")
# Функція для перезавантаження комп'ютера
def restart_computer():
    try:
        # Для Windows
        subprocess.run(["shutdown", "/r", "/t", "1"], check=True)
        speak("Комп'ютер перезавантажується.")
    except Exception as e:
        speak(f"Не вдалося перезавантажити комп'ютер: {e}")
def process_command(command):
    command = command.lower()
    if "вимкни комп'ютер" in command:
        shutdown_computer()
    elif "перезавантаж комп'ютер" in command:
        restart_computer()
    else:
        speak("Команда не розпізнана")
```

Рисунок 3.17 – Функція для вимикання та перезавантаження комп’ютера

Для створення зручного та простого інтерфейс користувача інтелектуальної системи супроводжувача користувача КП було використано бібліотеку tkinter (рис. 3.18).

3.4 Опис інтерфейсу

Інтерфейс голосового асистента був реалізований за допомогою бібліотеки tkinter, що дозволяє зручно взаємодіяти з користувачем. Головне вікно асистента має елемент керування, який дозволяє виконувати голосові команди для взаємодії з операційною системою (рис. 3.19).

Якщо натиснути на кнопку «*Слухати запит користувача*», система почне прослуховування голосових команд та виконувати функції які їй надасть користувач.

```
# Створення головного вікна
root = tk.Tk()
root.title("Голосовий асистент")

# Встановлення розміру
root.geometry("500x400")

# Поле для відображення повідомлень асистента та користувача
text_display = tk.Text(root, width=60, height=10, state=tk.DISABLED)
text_display.pack(pady=10)

# Текстове поле для вводу команди
entry_command = tk.Entry(root, width=50)

# Кнопка для обробки команд
button_process = tk.Button(root, text="Обробити", command=process_command)

# Кнопка для голосового вводу
button_voice = tk.Button(root, text="Голосова команда", command=process_command)

# Привітання користувача при запуску
greet_user()

# Розміщення елементів у вікні
text_display.pack(pady=10)
entry_command.pack(pady=5)
button_process.pack(pady=5)
button_voice.pack(pady=5)

# Функція для закриття додатку
def on_close():
    root.destroy()

root.protocol("WM_DELETE_WINDOW", on_close)

# Запуск головного вікна
root.mainloop()
```

Рисунок 3.18 – Створення інтерфейсу користувача

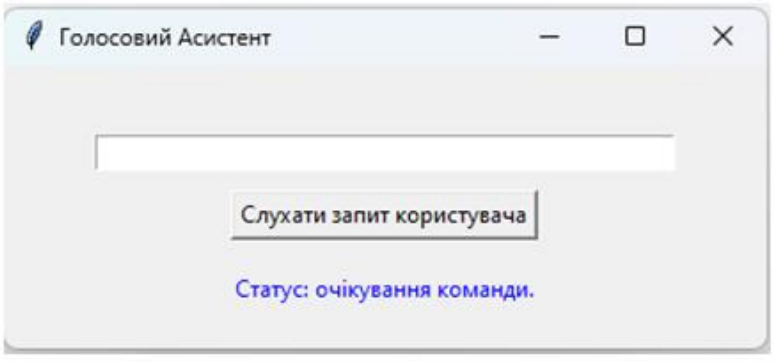


Рисунок 3.19 – Вікно для спілкування з користувачем голосового асистента

На рисунку 3.20 відображено, що користувач сказав «*Відкрити Google*» та система одразу відкрила у браузері відповідну сторінку пошукової системи.

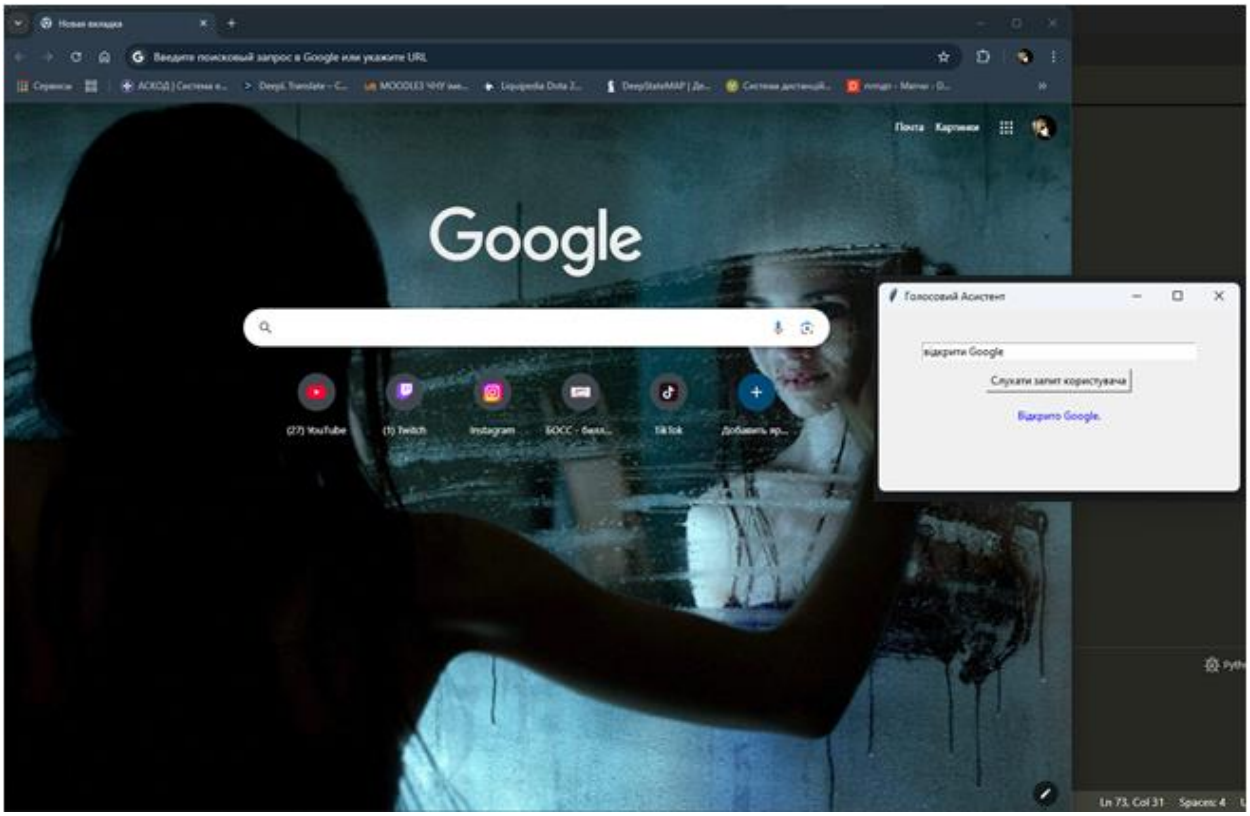


Рисунок 3.20 – Відкриття у браузері вікна Google за запитом користувача

На наступному рисунку 3.21 показано, як помічний за голосовим запитом: «*Знайти в гугл інформацію про Чорноморський національний університет ім. Петра Могили*» знаходить в Google відповідну інформацію.

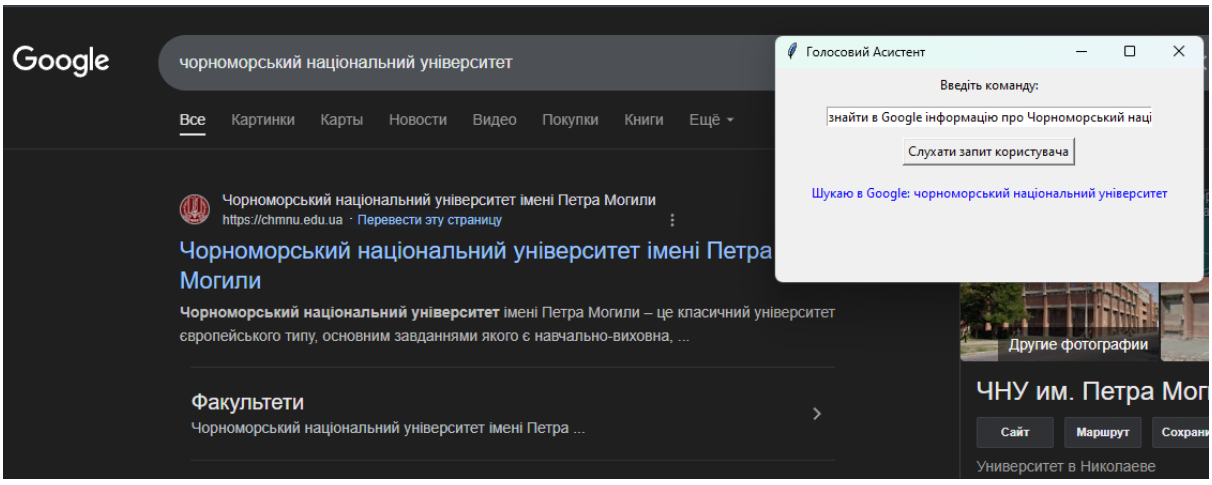


Рисунок 3.21 – Знайдена інформація про ЧНУ ім. П. Могили за запитом користувача

На рисунку 3.22 показано, як помічник за голосовим запитом: «*Знайти в Youtube канал Чорноморський національний університет*», знаходить відповідну інформацію.

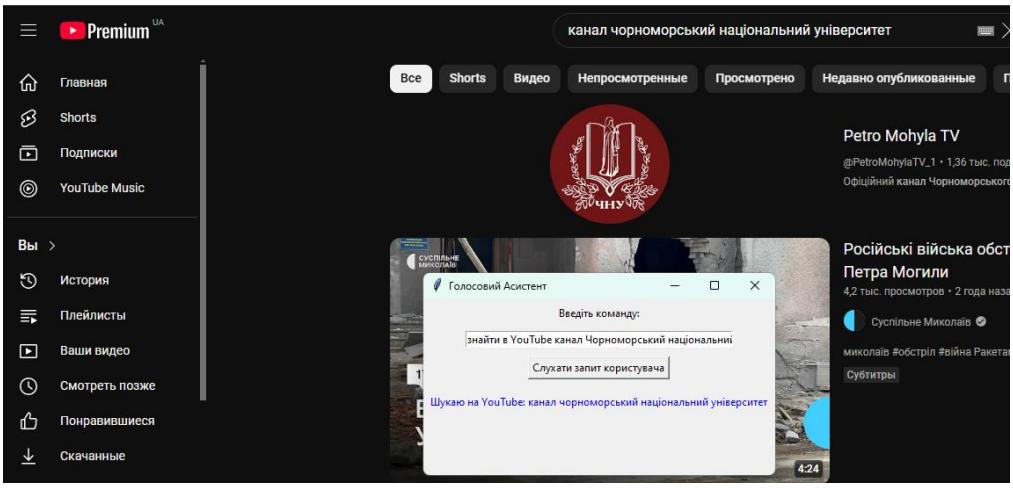


Рисунок 3.22 – Знайдені відео в Youtube за запитом

На рисунку 3.23 продемонстровано, що асистент користувача ПК може відкривати диски на комп’ютері, якщо користувач створить відповідний запит «Відкрити диск С».

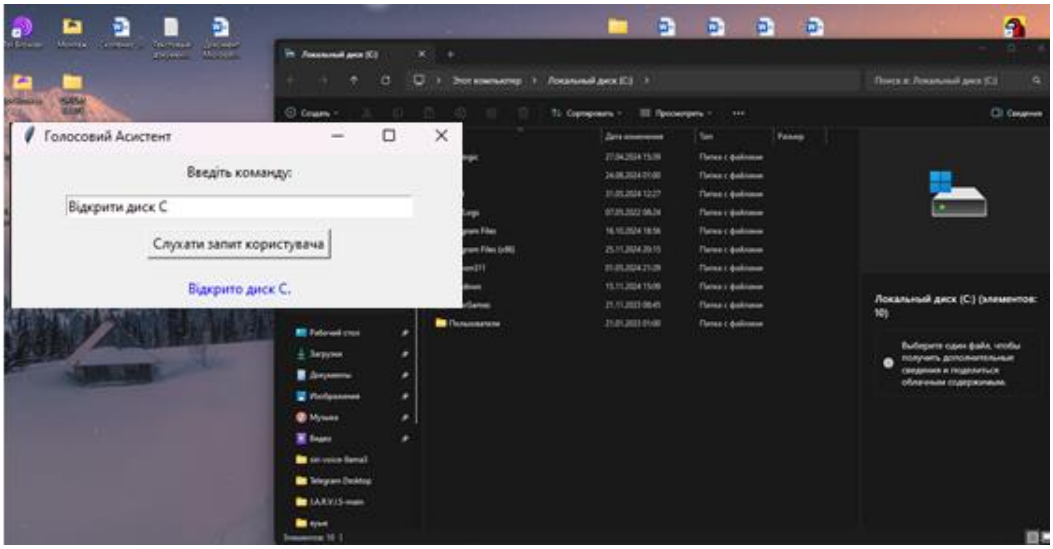


Рисунок 3.23 – Знайдений диск С за запитом

На рисунку 3.24 показано, як за запитом користувача «Відкрити папку Intel» здійснюється відкриття папки, яка знаходиться у вікні уже відкритого диску.

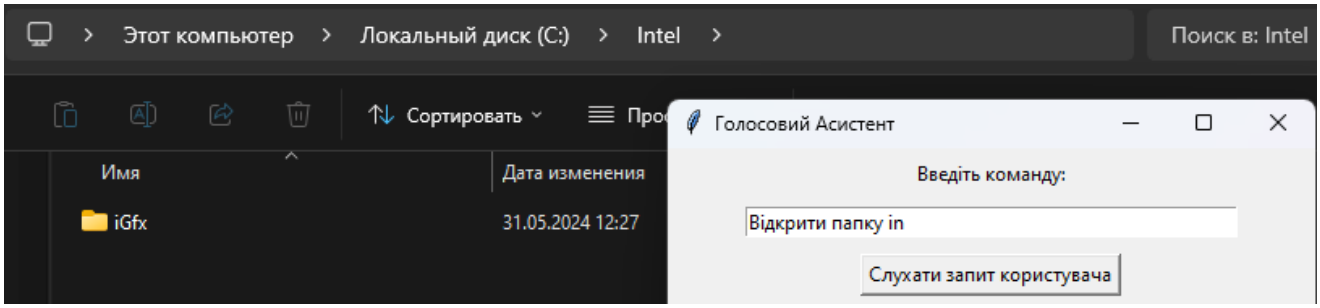


Рисунок 3.24 – Відкриття папки за запитом

На рисунках 3.25 та 3.26 показано відкриття файлів документів за запитом користувача із вказівкої їх назви.

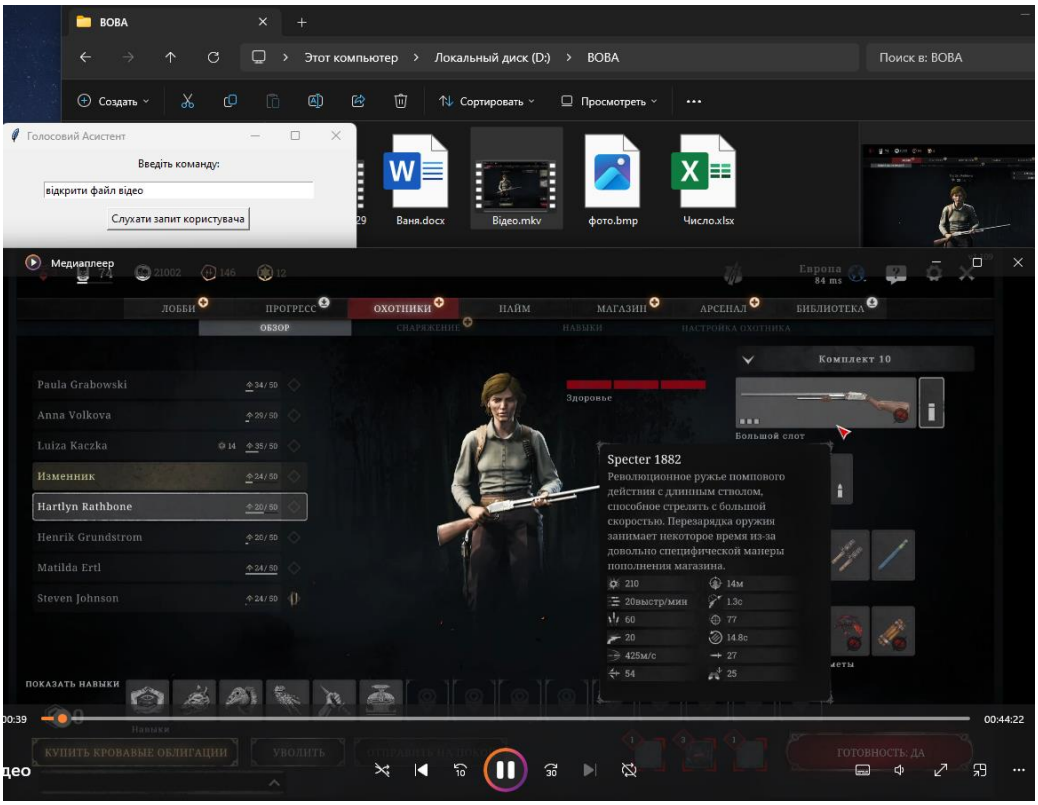


Рисунок 3.25 – Відкритий файл відео в папці D за запитом

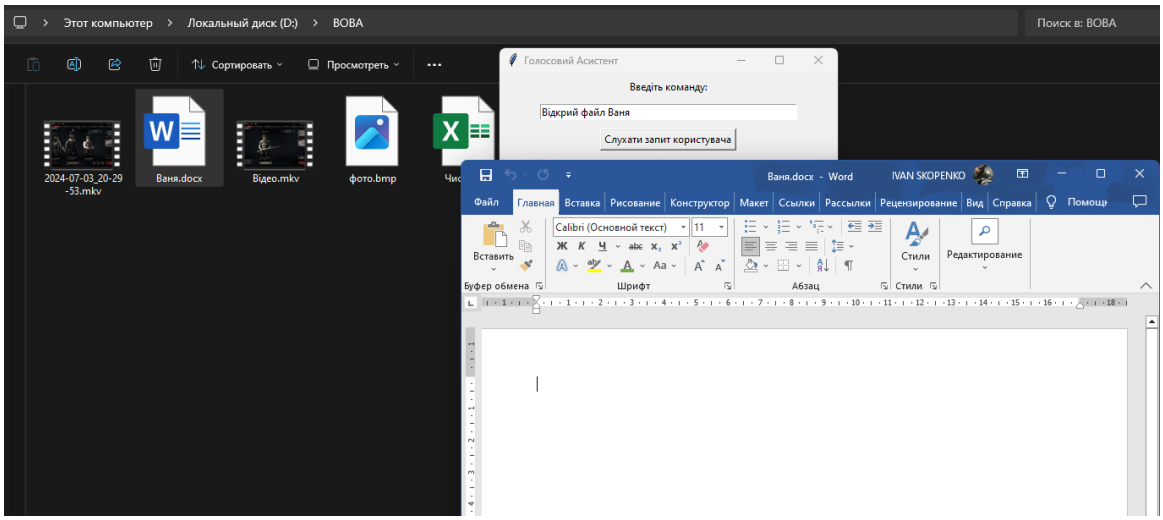


Рисунок 3.26 – Відкритий файл Word в папці D за запитом

За запитом користувача голосовий асистент може дати відповіді на питання, які стосуються поточного часу або погоди у своєму місті тощо (рис. 3.27, рис. 3.28). Що є досить важливим для користувачів із обмеженими функціональними можливостями.

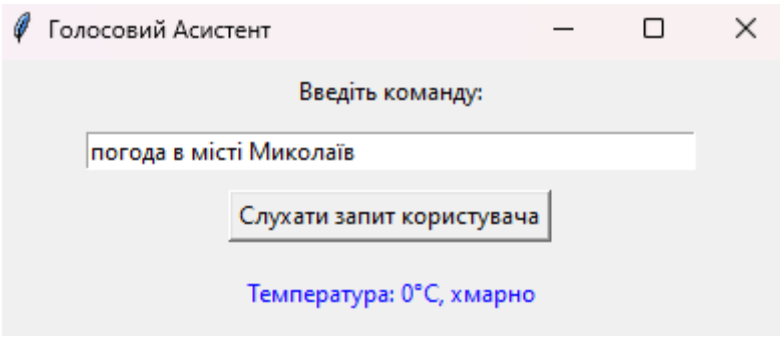


Рисунок 3.27 – Відповідь на запит про погоду

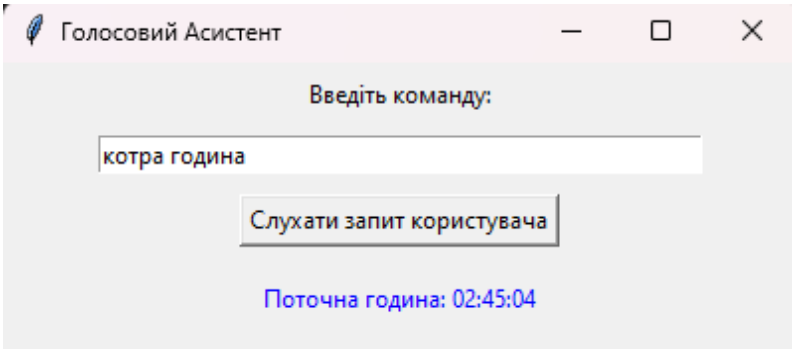


Рисунок 3.28 – Відповідь на запит про поточний час

На рисунку 3.29 продемонстровано, що асистент може відкривати програми на комп’ютері за голосовим запитом користувача на українській мові.

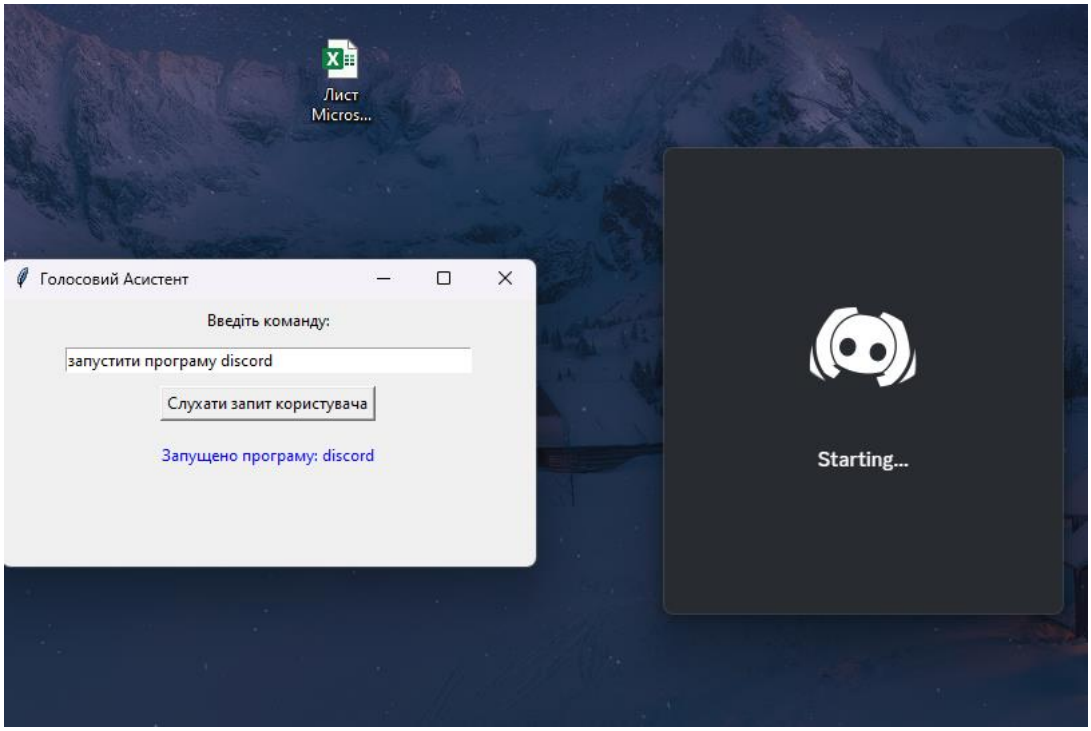


Рисунок 3.29 – Відкриття програми за запитом

Голосовий помічник дозволяє відкривати установлені на ПК програми для створення офісних документів: MS Word, MS Excel, MS PowerPoint (рис. 3.30-3.32).

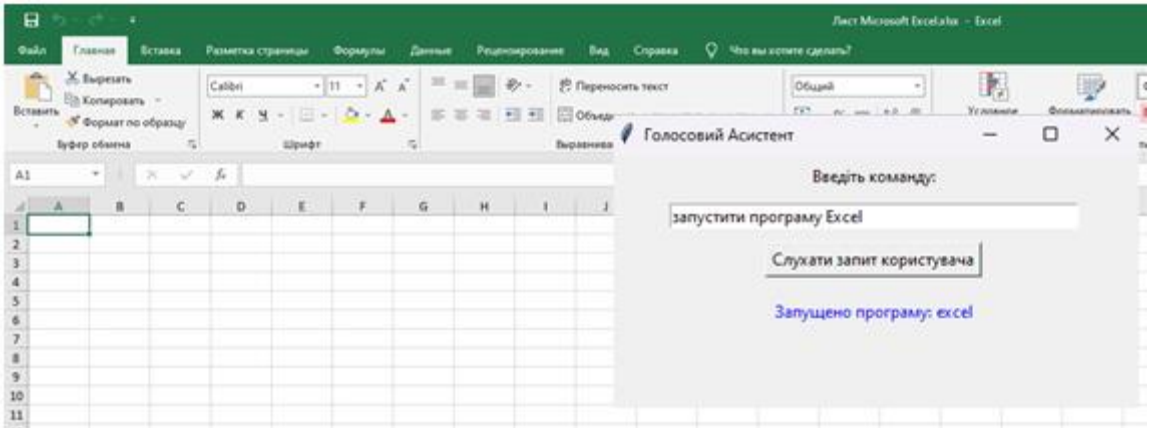


Рисунок 3.30 – Відкриття вікна програми MS Excel

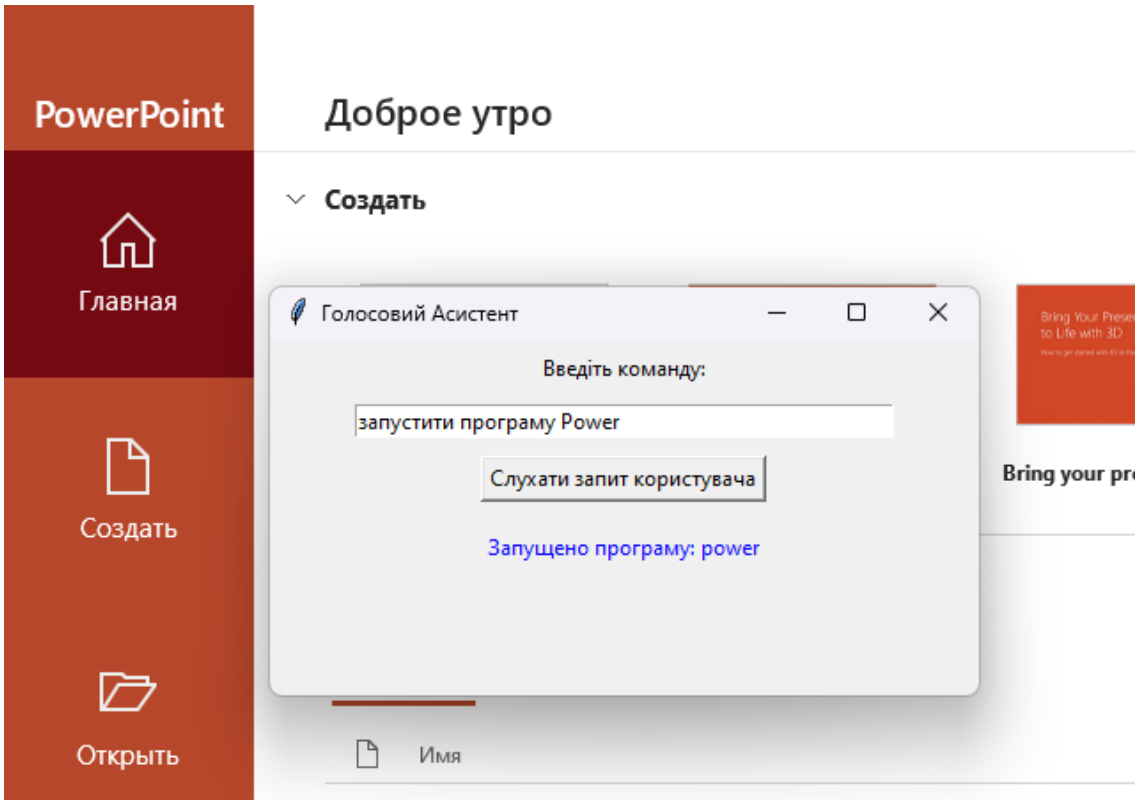


Рисунок 3.31 – Відкриття вікна програми MS PowerPoint

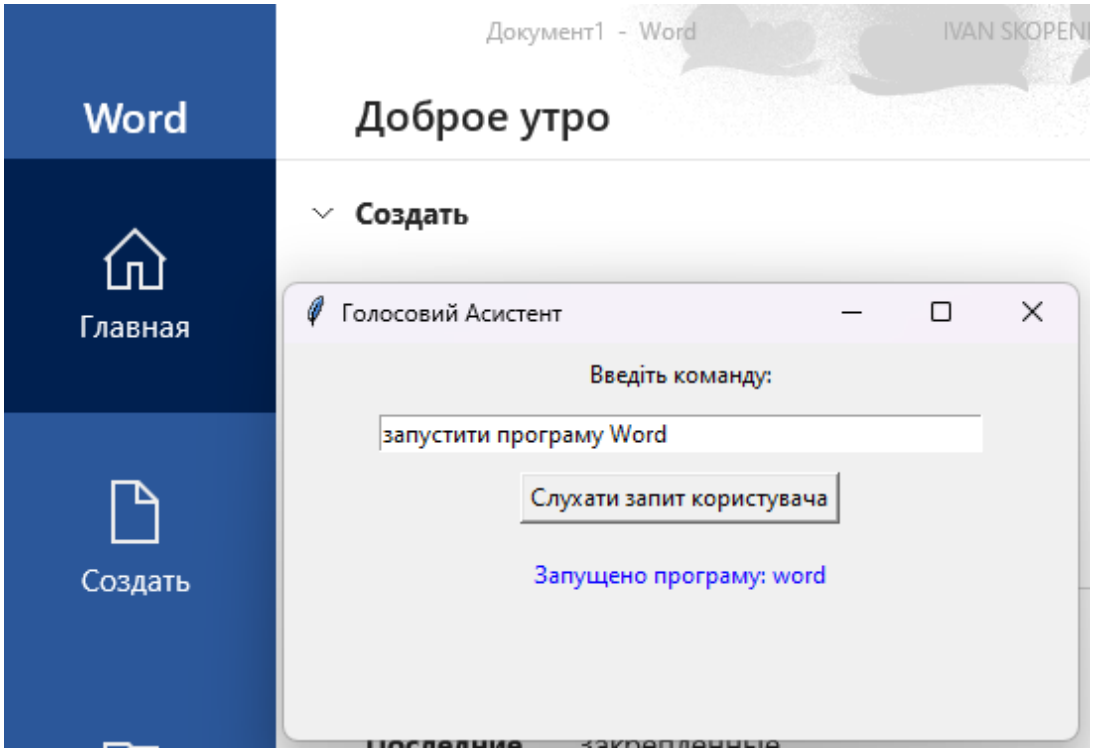


Рисунок 3.32 – Відкриття вікна програми MS Word

Також асистент може вимикати, перезавантажувати та переводити у режим сну комп’ютер за запитом користувача.

3.5 Тестування системи

Для дослідження якості розробленої системи було проведено тестування на літніх людях, які мають обмежений досвід роботи з ПК. Тестування включало оцінку зручності користування, ефективності взаємодії та точності виконання голосових команд.

Основною метою тестування було оцінити, наскільки інтерфейс голосового асистента доступний і зрозумілий для людей, що не мають досвіду в роботі з комп’ютерами, а також перевірити, чи система правильно реагує на прості голосові команди, які надаються.

Для тестування було обрано кілька літніх людей різного віку та з різним рівнем інформаційної підготовки. Вони мали різні навички: від тих, хто лише іноді користується комп'ютером, до тих, хто зовсім не знайомий з комп’ютерними технологіями.

Тестування проводилось на десктопному комп’ютері з голосовим інтерфейсом. Кожен учасник мав можливість спілкуватися з голосовим асистентом, видаючи йому прості голосові команди. Програма була налаштована так, щоб виконувати базові операції, такі як відкриття програм, відкриття папок і виконання простих завдань. Кожному учаснику було запропоновано виконати кілька простих завдань за допомогою голосового асистента:

– відкрити певну програму;

– відкрити папку на комп'ютері;

– виконати пошук певних файлів або програм через голосовий запит;

– попросити асистента виконати певну дію, наприклад, зачитати текст чи нагадати подію.

Більшість учасників відзначили, що їм було зручно взаємодіяти з системою через голосові команди. Незважаючи на відсутність досвіду, більшість користувачів змогли зрозуміти основні принципи взаємодії з асистентом без додаткових пояснень. Завдання були виконані за короткий час, більшість учасників змогли отримати відповідь або виконати операцію в межах 5–10 секунд після подання голосової команди.

Система правильно реагувала на більшість голосових команд, що були чітко висловлені. Деякі учасники зауважили труднощі у вимові складних слів або фраз, але в таких випадках система намагалася правильно виконати команду, навіть якщо була певна неточність у вимові.

Розроблена інтелектуальна система показала свою ефективність у використанні серед людей, які мають обмежений досвід роботи з комп’ютерами. Система забезпечує простоту взаємодії за допомогою голосових команд, що робить її доступною для людей з слабкою інформаційною підготовкою та обмеженими фізичними можливостями. Тестування виявило деякі дрібні проблеми з розпізнаванням деяких слів, однак загальна оцінка системи була позитивною.

Таке тестування є важливим етапом у розробці технологій, орієнтованих на користувачів з обмеженим досвідом роботи з сучасними комп'ютерними технологіями. На основі отриманих результатів можна провести подальшу оптимізацію системи для підвищення її зручності і точності у виконанні голосових команд.

Висновки до розділу 3

У даному розділі описано моделювання, розробку та програмну реалізацію інтелектуальної системи супроводження користувача ПК з використанням нейромережевих моделей обробки природної мови. Донавчання мовної моделі LlaMA здійснювалося з використанням фреймворку LlamaIndex й архітектури RAG

на створеному DataSet текстового формату JSON, який охоплює типові команди, що надходять від користувачів ПК.

Описано програмну реалізацію та описано інтерфейс системи супроводження користувача ПК на основі великих мовних моделей, архітектура яких базується на нейромережевих моделях трансформерів.

Розроблена система має значні соціальні та технологічні переваги за рахунок полегшення роботи з ПК, зокрема:

– підвищення доступності та зручності роботи з комп'ютером для ветеранів війни та людей з обмеженими можливостями;

– автоматизація рутинних завдань, таких як керування файлами, пошук інформації, запуск програм, що дозволить користувачам зосередитися на важливіших аспектах роботи;

– покращення точності і швидкості виконання завдань через персоналізацію інтерфейсу та інтерактивних елементів;

– забезпечення постійної допомоги та підказок, які допоможуть користувачу ефективніше використовувати ПК та уникати помилок.

Розроблена система має зручний інтерфейс, зрозумілий для кожного користувача, на головній сторінці є поле для відображення отриманих голосовим помічником запитів, його відповідей у разі потреби та кнопка для початку спілкування. Систему протестовано, перевірено коректність виконання команд користувацького інтерфейсу у відповідь на запити клієнтів.

ВИСНОВКИ

Проведене дослідження дозволяє зробити наступні висновки. Використання в підтримці користувачів ПК нейромережевих мовних моделей сприяє розв'язанню актуальних завдань підтримки користувачів з обмеженими фізичними можливостями та слабким рівнем інформаційної підготовки у їх роботі з ПК.

Досліджено теоретичні засади підтримки UI-інтерфейса користувача в операційній системі Windows із використанням нейромережевих моделей обробки природної мови. Установлено, що революційні зміни в системах обробки природної мови обумовлені виникненням великих мовних моделей (LLM), архітектура яких базується на трансформерах – великих наборах нейронних мереж із можливістю самонавчання. Архітектура трансформерів дозволяє використовувати дуже великі нейронні моделі з сотнями мільярдів параметрів і підтримкою багатьох мов. Здійснений огляд нейромережевих моделей для обробки природної мови дозволив установити, що застосування передових технологій обробки природної мови – нейромережевих моделей на основі трансформерів, дозволяє створити віртуальний помічник, здатний виконувати аудіокоманди операційної системи з високою точністю, що робить їх ідеальними для системи підтримки користувача ПК у операційній системі Windows. Серед LLM-моделей було виділено модель LlaMA, яка є моделлю NLP з мільярдами параметрів, навчена 20 мовах, має оптимізовану продуктивність, реалізована з використанням більш досконалих алгоритмів токенізації й може бути реалізована у інтелектуальних системах допомоги для виконання контекстних та точних дій у відповідь на аудіозапити користувачів.

У контексті інтелектуальних систем супроводження користувача нейромережі дозволяють: автоматично вивчати індивідуальний стиль роботи користувача та підлаштовувати інтерфейс під його потреби; прогнозувати можливі дії користувача та надавати йому відповідні підказки чи допомогу; забезпечувати безперервну адаптацію системи до змін у поведінці чи стані користувача, що особливо важливо для осіб, які мають обмежені фізичні та ментальні можливості.

Це дозволяє значно спростити взаємодію з комп'ютером для людей, які мають труднощі у його використанні, та підвищує їхню продуктивність.

Здійснений аналіз існуючих програмних рішень голових асистентів (Siri, Google Assistant, Amazon Alexa, Microsoft Cortana) дозволив установити, що голосова підтримка більшою мірою реалізована у мобільних операційних системах. Їх функціональність для підтримки користувача операційної системи ПК є досить обмеженою. Більшість з них є безкоштовними, а вартість якісних продуктів є високою і для українських користувачів ще й недоступність спілкування з голосовими помічниками українською мовою. Тому є потреба у створення системи підтримки користувача ПК із використанням методів обробки природної української мови на основі нейромережевих моделей.

Для створення інтелектуальної системи підтримки користувача ПК було обрано мову програмування Python, середовище розробки Visual Studio Code. Для автоматизації роботи з графічним інтерфейсом операційної системи Windows обрано Python-бібліотеки os, webbrowser, Tkinter, PyAutoGUI. Бібліотека os надає безліч функцій, які дозволяють нам взаємодіяти з базовою операційною системою, включаючи доступ та керування файловою системою, змінними середовищами і процесами. Webbrowser надає високорівневий інтерфейс для відображення та перегляду вебсторінок. Tkinter дозволяє створити повноцінний працюючий інтерфейс користувача мобільного або десктопного застосунку. Бібліотека PyAutoGUI має багато засобів для автоматизації основних операцій з графічним інтерфейсом операційної системи – управління мишею, клавіатурою, роботи з вікнами та зображеннями.

Для обробки голосових команд користувача було використано бібліотеки Pocketsphinx, SpeechRecognition, Pyttsx3, Hugging Face, Transformers. Pocketsphinx застосовано для реалізації голосового управління. SpeechRecognition використана для розпізнавання голосових команд, які дозволяють взаємодіяти з системою. Бібліотека pyttsx3 дозволяє озвучувати повідомлення, підказки або відповіді на

запити користувача. Transformers, підтримувана Hugging Face, використана для доступу до колекції навчаних нейромережевих моделей оброки природної мови.

Хмарний сервіс Google Cloud Speech API обрано для автоматичного розпізнавання мови в реальному часі. Для полегшення роботи з мовною моделлю LlaMA вирішено використовувати фреймворк Llamaindex.

Здійснено моделювання, розробку та програмну реалізацію інтелектуальної системи супроводження користувача ПК з використанням нейромережевих моделей обробки природної мови. Донавчання мовної моделі LlaMA здійснювалося з використанням фреймворку LlamaIndex й архітектури RAG на створеному DataSet текстового формату JSON, який охоплює типові команди, що надходять від користувачів ПК.

Розроблена система має значні соціальні та технологічні переваги за рахунок полегшення роботи з ПК, зокрема:

– підвищення доступності та зручності роботи з комп'ютером для ветеранів війни та людей з обмеженими можливостями;

– автоматизація рутинних завдань, таких як керування файлами, пошук інформації, запуск програм, що дозволить користувачам зосередитися на важливіших аспектах роботи;

– покращення точності і швидкості виконання завдань через персоналізацію інтерфейсу та інтерактивних елементів;

– забезпечення постійної допомоги та підказок, які допоможуть користувачу ефективніше використовувати ПК та уникати помилок.

Розроблена система має зручний інтерфейс, зрозумілий для кожного користувача. Систему протестовано, перевірено коректність виконання команд користувацького інформфейсу у відповідь на запити клієнтів.

Поставлені завдання виконано у повному обсязі. У загальному висновку можна сказати, що дана система є актуальною на сьогоднішній деньпроте її функціональність можу бути розширена у подальшому за рахунок виконання більшої кількості команд UI-інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Amaratunga T. Understanding Large Language Models: Learning Their Underlying Concepts and Technologies 1st ed. Edition, 2023. – 176 p.

2. Що таке обробка природної мови (Natural Language Processing, NLP): вебсайт. URL: https://thetransmitted.com/adlucem/shho-take-obrobka-pryrodnoyi-movy-natural-language-processing-nlp/ (дата звернення: 15.10.2024).

3. Barrat J. Our Final Invention : Artificial Intelligence and the End of the Human Era, 2023. – 336 p.

4. Обробка природної мови (NLP) у Python (вкладання слів та семантична подібність): вебсайт. URL: https://oleg-dubetcky.medium.com/обробка-природної-мови-nlp-у-python-з-кодом-частина-6-вкладання-слів-та-семантична-подібність-2645d9c64122 (дата звернення: 15.09.2024).

5. What is LLM (Large Language Model): вебсайт. URL: https://aws.amazon.com/what-is/large-language-model/?nc1=h_ls (дата звернення: 29.09.2024).

6. Raschka. S. Build a Large Language Model (From Scratch), 2024. – 400 p.

7. Hagging Face, Transformers: вебсайт. URL: https://huggingface.co/docs/transformers/index (дата звернення: 08.10.2024).

8. Radford A., Narasimhan K., Salimans T., Sutskever I., et al. Improving language understanding by generative pre-training, OpenAI, 2018.

9. H. Touvron, T. Lavril, G. Izacard and all. Llama: Open and Efficient Foundation Models. Meta IA, 2023: вебсайт. URL: https://arxiv.org/abs/2302.13971 (дата звернення: 18.10.2024).

10. The Llama 3 Herd of Models. Llama Team, July 2024: вебсайт. URL: https://ar5iv.labs.arxiv.org/html/2407.21783 (дата звернення: 08.09.2024).

11. Better language models and their implications. OpenAI: вебсайт. URL: https://openai.com/index/better-language-models/ (дата звернення: 13.10.2024).

12. Code E. Tu. LLM Universe: Building LLMs, OpenAI & Llama 2: A Comprehensive Guide to Large Language Model Development, OpenAI and Meta Llama 2 Kindle Edition, 2024. – 351 p.

13. Meta Llama: все, що потрібно знати про відкриту генеративну AI-модель: вебсайт. URL: https://proit.ua/meta-llama-vsie-shcho-potribno-znati-pro-vidkritu-ghienierativnu-ai-modiel/ (дата звернення: 10.09.2024).

14. Llama: вебсайт. URL: https://ai.meta.com/blog/large-language-model-llama-meta-ai/ (дата звернення 01.10.2024).

15. Усе про нову модель штучного інтелекту Llama від Meta: вебсайт. URL: https://speka.media/use-pro-novu-model-stucnogo-intelektu-llama-vid-meta-pk40gw (дата звернення: 12.09.2024).

16. Beazley D., Jones B. Python Cookbook: Recipes for Mastering Python 3, 2023. – 704 p.

17. Автоматизація завдань з Python: найкращі практики та приклади: вебсайт. URL: https://foxminded.ua/avtomatyzatsiia-zavdan-z-python/ (дата звернення: 12.08.2024).

18. 10 найкращих бібліотек Python для GUI: вебсайт. URL: https://www.unite.ai/uk/10-найкращих-бібліотек-python-для-графічного-інтерфейсу/ (дата звернення: 06.10.2024).

19. OS – Miscellaneous operating system interfaces: вебсайт. URL: https://docs.python.org/3/library/os.html (дата звернення: 04.09.2024).

20. SpeechRecognition: вебсайт. URL: https://pypi.org/project/SpeechRecognition/ (дата звернення: 11.09.2024).

21. Hugging Face Transformers: вебсайт. URL: https://huggingface.co/docs/transformers/en/index (дата звернення: 9.09.2024).

22. Python Transformers: вебсайт. URL: https://pypi.org/project/transformers/ (дата звернення: 11.10.2024).

23. Framework For LLM Agents: вебсайт. URL: https://www.llamaindex.ai/framework (дата звернення: 22.08.2024).

24. Розуміння тонкого налаштування LLM: адаптація великих мовних моделей до ваших унікальних вимог: вебсайт. URL: https://www.unite.ai/uk/розуміння-llm-тонке-налаштування-адаптація-великих-мовних-моделей-до-ваших-унікальни% (дата звернення: 19.10.2024).

25. Llama_index documentation: вебсайт. URL: https://docs.llamaindex.ai/en/stable/ (дата звернення 01.09.2024).

26. Webbrowser – Convenient web-browser controlle: вебсайт. URL: https://docs.python.org/3/library/webbrowser.html (дата звернення: 27.09.2024).