Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

За освітнім ступенем «бакалавр»

На тему:

РОЗРОБКА ЧАТ-БОТА ДЛЯ АВТОМАТИЗОВАНОГО НАВЧАННЯ КОРИСТУВАЧІВ З ВИКОРИСТАННЯ КОНТЕКСНИХ МОДЕЛЕЙ

Виконав:
здобувач IV курсу
групи ІПЗ-41
спеціальності 121 «Інженерія програмного забезпечення»
Доронін Віталій Олегович
Науковий керівник:
к.т.н., доцент Гаврилюк В.І.

Рівне – 2023

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі інформаційних технологій автоматизація навчальних процесів є однією з ключових тенденцій у галузі освіти. З кожним роком зростає кількість користувачів, які віддають перевагу онлайн-навчанню, що робить необхідним створення нових, більш ефективних засобів та інструментів для забезпечення якісного навчального процесу. Одним з таких інструментів є чат-боти.

Чат-боти для навчання надають можливість забезпечити інтерактивний, персоналізований підхід до навчання, який адаптується до індивідуальних потреб кожного користувача. Використання контекстних моделей, таких як GPT-4, дозволяє створювати чат-боти, які можуть ефективно обробляти та аналізувати великі обсяги інформації, забезпечуючи високий рівень розуміння і відповідності контексту під час навчального процесу.

Актуальність теми також зумовлена стрімким розвитком технологій штучного інтелекту та машинного навчання. Впровадження цих технологій у навчальний процес дозволяє значно підвищити його ефективність, знизити витрати на навчання та забезпечити доступ до якісної освіти для широкого кола користувачів.

Крім того, у зв'язку з глобальною пандемією COVID-19, яка спричинила значні зміни в освітніх процесах, потреба в дистанційних навчальних технологіях стала ще більш актуальною. Чат-боти можуть допомогти у вирішенні проблем, пов'язаних з дистанційним навчанням, забезпечуючи постійну підтримку та супровід студентів.

Таким чином, розробка чат-бота для автоматизованого навчання користувачів з використанням контекстних моделей є надзвичайно актуальною темою, яка відповідає сучасним вимогам і тенденціям розвитку освіти та інформаційних технологій.

Мета дослідження - розробка чат-бота для автоматизованого навчання користувачів з використанням технологій Dialogflow та GPT-3.5, який забезпечуватиме ефективну і персоналізовану підтримку у навчальному процесі. Основні завдання, що стоять перед розробкою, включають створення системи, яка зможе надавати коректні, контекстуально релевантні відповіді на запити користувачів, а також адаптуватися до індивідуальних потреб та рівня знань кожного користувача.

Для досягнення цієї мети передбачено реалізацію наступних **завдань**:

1. Аналіз існуючих технологій та інструментів для розробки чат-ботів, зокрема Dialogflow та GPT-3.5.
2. Визначення основних вимог до функціоналу чат-бота, який буде використовуватися для навчання.
3. Розробка архітектури чат-бота, яка інтегрує можливості Dialogflow для обробки природної мови та GPT-3.5 для генерації відповідей на запити користувачів.
4. Реалізація алгоритмів, що забезпечують обробку та аналіз запитів користувачів, а також формування відповідей на основі контексту.
5. Тестування та оцінка ефективності розробленого чат-бота у навчальному процесі.
6. Оптимізація та вдосконалення чат-бота на основі результатів тестування та відгуків користувачів.

Відповідно до мети було поставлено наступні **завдання:**

- Провести збір і аналіз інформації, яку надають користувачі чат-боту під час спілкування.
- Створити модель машинного навчання, яка буде навчатися на основі зібраних даних користувачів.

- Використовуючи навчену модель, покращити роботу чат-бота, забезпечивши більш точні та релевантні відповіді на запитання користувачів.

Об’єкт дослідження – Процес розробки та впровадження чат-бота з використанням платформ Dialogflow та OpenAI API.

Предмет дослідження – Методи машинного навчання та контекстні моделі у роботі чат-бота.

Методи дослідження:

1. **Аналіз літератури та наукових статей** - для визначення актуальних методів розробки та використання чат-ботів.
2. **Розробка програмного забезпечення** - для створення та інтеграції чат-бота з платформами Dialogflow та OpenAI API.
3. **Тестування та оптимізація** - для перевірки функціональності та ефективності роботи чат-бота.
4. **Емпіричні дослідження** - збір та аналіз даних від користувачів для подальшого навчання та покращення роботи чат-бота.

Практичне значення дослідження.

Розроблений чат-бот може бути використаний для автоматизованої підтримки користувачів, навчання нових співробітників або клієнтів, а також для інтерактивного обслуговування клієнтів у різних сферах. Використання платформ Dialogflow та OpenAI API забезпечує високу точність та релевантність відповідей, що значно підвищує ефективність взаємодії з користувачами.

Апробація і впровадження результатів дослідження.

Основні результати роботи доповідалися на студентській конференції (Рівне, 16 травня 2024 р.).

Структура роботи. Дипломна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи – 50 сторінок**.**

РОЗДІЛ 1. ЧАТ-БОТИ

1.1. Визначення і типи

Чат-боти, або "чат-роботи", представляють собою програмні засоби, які використовують штучний інтелект (ШІ) для симуляції людської розмови. Вони можуть функціонувати як через текстові, так і голосові інтерфейси, забезпечуючи користувачів інформацією, виконуючи різноманітні завдання або надаючи підтримку у вирішенні певних проблем. Чат-боти можуть бути інтегровані в різноманітні платформи, такі як вебсайти, мобільні додатки та месенджери.

Чат-боти класифікуються за різними критеріями, серед яких найважливішими є функціональні можливості та методи реалізації:

1. **За функціональними можливостями**:
   - **Питально-відповідні чат-боти**: Призначені для відповіді на конкретні запитання користувачів, зазвичай базуються на базі даних відповідей.
   - **Транзакційні чат-боти**: Виконують певні завдання, такі як бронювання квитків, замовлення товарів або послуг.
   - **Соціальні чат-боти**: Створені для підтримання розмови на загальні теми, зазвичай для розваги або навчання.
2. **За методами реалізації:**
   - **Скриптові чат-боти:** Працюють за заздалегідь визначеними сценаріями, які визначають послідовність дій та відповідей на запити користувача.
   - **Чат-боти на базі машинного навчання:** Використовують алгоритми машинного навчання та обробки природної мови (NLP) для аналізу та відповіді на запити користувачів.

Наразі чат-боти мають дуже велике застосування у багатьох галузях, зокрема:

- **Клієнтська підтримка**: Автоматизація відповідей на запити клієнтів, що допомагає зменшити навантаження на персонал. Наприклад, компанія "Amazon" використовує чат-бота для підтримки клієнтів, який може автоматично відповідати на запити про статус замовлення, обробляти повернення та обмін товарів, а також надавати інформацію про продукти.
- **Електронна комерція**: Допомога у виборі товарів, оформленні замовлень та інформуванні про статус доставки. Для прикладу, магазин одягу "H&M" інтегрував чат-бота у свою систему продажів, що допомагає клієнтам вибирати одяг на основі їхніх вподобань та надає рекомендації щодо стилю. Бот також може допомогти з оформленням замовлення та надавати інформацію про знижки та акції.
- **Освіта**: Інтерактивні платформи для навчання та надання консультацій студентам. Наприклад, платформа для онлайн-навчання "Coursera" використовує чат-бота для консультування студентів з приводу вибору курсів, реєстрації на заняття та надання інформації про програми навчання. Бот також може відповідати на запитання щодо технічної підтримки та допомагати у вирішенні проблем.
- **Охорона здоров'я**: Надання медичних консультацій, відстеження стану здоров'я та підтримка пацієнтів. Для прикладу, медичний додаток "Babylon Health" використовує чат-бота, який може надавати користувачам попередні медичні консультації, оцінюючи симптоми та пропонуючи можливі діагнози. Бот також може допомогти з записом на прийом до лікаря та надавати інформацію про ліки.

1.2. Принцип роботи чат-ботів

Для початку про архітектуру чат-ботів. Зазвичай вони складаються з кількох ключових компонентів:

- **Інтерфейс користувача (UI)**: Платформа, через яку користувач взаємодіє з ботом (месенджери, вебсайти, мобільні додатки).
- **Обробка природної мови (NLP)**: Компонент, який перетворює текст або голос користувача в зрозумілі для бота команди.
- **Модуль управління діалогом**: Визначає логіку та сценарії ведення розмови.
- **База знань**: Містить інформацію, на основі якої бот надає відповіді або виконує дії.
- **Інтеграція з зовнішніми системами**: Здійснює обмін даними з іншими програмами та сервісами для виконання транзакцій або надання додаткової інформації.

Чат-боти виконують взаємодію з користувачами через серію етапів, які включають обробку природної мови (NLP), аналіз даних і генерацію відповідей.

1.3. Оброблення мови

Обробка природної мови (Natural Language Processing, NLP) є ключовим елементом у роботі чат-ботів, що дозволяє їм розуміти та генерувати текст на природній мові. Цей процес складається з кількох послідовних етапів, кожен з яких відіграє важливу роль у забезпеченні точності та ефективності роботи чат-бота.

1. Аналіз вхідних даних

Перший етап обробки природної мови полягає в аналізі вхідних даних, що можуть бути представлені як текстові або голосові повідомлення від користувача.

- **Розпізнавання мови (ASR)**: У випадку голосових чат-ботів, застосовується автоматичне розпізнавання мови (Automatic Speech Recognition, ASR), що перетворює голосові сигнали в текстовий формат. Використовуються алгоритми обробки сигналів та моделі машинного навчання для точного розпізнавання слів.

- **Попередня обробка тексту**: Текстові дані проходять через етап попередньої обробки, який включає нормалізацію тексту (перетворення всіх символів у нижній регістр, видалення зайвих пробілів), очищення від шуму (видалення знаків пунктуації, спеціальних символів), а також виправлення орфографічних помилок.

2. Токенізація

Токенізація - це процес поділу тексту на окремі елементи, які називаються токенами. Токени можуть бути словами, фразами або реченнями.

- **Словова токенізація**: Текст розбивається на окремі слова. Це один із найпоширеніших типів токенізації, оскільки слова є основними одиницями аналізу.
- **Фразова токенізація**: Текст розбивається на фрази або словосполучення, що може бути корисним для розуміння контексту.

3. Лематизація та стемінг

Лематизація та стемінг є методами нормалізації тексту, що дозволяють зменшити кількість унікальних слів, які потрібно аналізувати.

- **Лематизація**: Процес приведення слів до їх базової або початкової форми (леми). Наприклад, слова "бігти", "біжить" та "біг" будуть приведені до форми "бігти".
- **Стемінг**: Процес скорочення слів до їх кореневої форми або основи (стема). Наприклад, слова "running" та "runner" будуть скорочені до "run".

4. Аналіз синтаксису та семантики

На цьому етапі аналізується структура речень (синтаксис) та значення слів і фраз (семантика).

- **Синтаксичний аналіз**: Визначає граматичну структуру речення, встановлюючи відносини між словами. Це допомагає виявити підмет, присудок, додатки та інші частини речення.

- **Семантичний аналіз**: Зосереджується на значенні слів і фраз у контексті речення. Це дозволяє боту зрозуміти сенс запиту користувача.

5. Визначення намірів та сутностей

Чат-бот повинен розпізнати наміри (інтенції) користувача та сутності (ключові елементи запиту).

- **Визначення намірів**: Використовуються моделі машинного навчання для класифікації запиту користувача за певними категоріями намірів. Наприклад, наміри можуть включати запит на інформацію, здійснення бронювання або оформлення замовлення.
- **Визначення сутностей**: Виявляються ключові елементи запиту, такі як дати, імена, місця або інші важливі дані, що допомагають уточнити намір користувача. Це може включати витягування іменованих сутностей (NER - Named Entity Recognition).

6. Генерація відповіді

На основі визначених намірів та сутностей чат-бот генерує відповідь.

- **Формування відповіді**: Відповідь може бути згенерована динамічно або взята з заздалегідь підготовленої бази знань. У випадку динамічної генерації використовуються моделі генерації природної мови (NLG - Natural Language Generation).
- **Виконання дії**: Якщо запит користувача передбачає виконання певної дії (наприклад, бронювання квитка), бот взаємодіє з відповідними зовнішніми системами для виконання цієї дії.

7. Зворотний зв'язок та навчання

Чат-боти постійно вдосконалюються завдяки зворотному зв'язку від користувачів та постійному навчанню.

- **Збір зворотного зв'язку**: Користувачі можуть оцінювати відповіді бота, що дозволяє виявляти помилки та недоліки в роботі.

- **Машинне навчання**: Використовується для постійного вдосконалення моделей, що аналізують та генерують відповіді, на основі зібраних даних та зворотного зв'язку.

РОЗДІЛ 2. МАШИННЕ НАВЧАННЯ ТА КОНТЕКСТНІ МОДЕЛІ

2.1. Машинне навчання

Машинне навчання (ML) є підгалуззю штучного інтелекту (AI), яка фокусується на розробці алгоритмів і статистичних моделей, що дозволяють комп'ютерам навчатися та приймати рішення без явного програмування. Це наукова дисципліна, що об'єднує елементи математики, статистики, інформатики та теорії керування. За останні десятиліття машинне навчання зазнало значного розвитку та стало невід'ємною частиною багатьох галузей.

Машинне навчання базується на ідеї, що системи можуть навчатися з даних, ідентифікувати закономірності та приймати рішення з мінімальним втручанням людини. Процес машинного навчання включає три основні етапи:

1. **Збір даних**: Накопичення великої кількості даних, які будуть використовуватися для навчання моделі.
2. **Підготовка даних**: Обробка та очищення даних, приведення їх до вигляду, придатного для навчання.
3. **Навчання моделі**: Використання алгоритмів для створення моделі, що може передбачати або класифікувати нові дані.

2.2. Використання машинного навчання

Машинне навчання знаходить застосування у різноманітних галузях, від медицини до фінансів, від промисловості до розваг. Основні напрямки його використання включають:

1. **Розпізнавання образів**: Використовується у системах безпеки, медицині (наприклад, діагностика раку на основі зображень МРТ), та автомобільній промисловості (системи автономного водіння).

2. **Обробка природної мови (NLP)**: Застосовується в чат-ботах, системах перекладу, аналізі настроїв, а також у пошукових системах.
3. **Рекомендаційні системи**: Використовуються в інтернет-магазинах (Amazon, eBay), платформах потокового відео (Netflix, YouTube) для надання персоналізованих рекомендацій.
4. **Фінансовий аналіз**: Включає прогнозування ринкових трендів, виявлення шахрайства, управління ризиками.
5. **Медична діагностика**: Використовується для прогнозування захворювань, аналізу медичних записів, розробки нових лікарських засобів.

Ось декілька більш конкретних прикладів використання:

1. **Автономні транспортні засоби**: Компанії, такі як Tesla, Google (Waymo), використовують машинне навчання для створення систем автопілоту, які здатні орієнтуватися у дорожніх умовах, розпізнавати об'єкти та приймати рішення в реальному часі.
2. **Розпізнавання мови**: Apple Siri, Amazon Alexa, та Google Assistant використовують алгоритми машинного навчання для розпізнавання та обробки команд природною мовою, надаючи користувачам можливість взаємодіяти з пристроями за допомогою голосу.
3. **Аналіз великих даних**: Компанії, такі як IBM та Microsoft, використовують машинне навчання для аналізу великих обсягів даних, що дозволяє виявляти тренди та закономірності, які недоступні традиційними методами.

2.3. Переваги та недоліки машинного навчання

Перевагами можна назвати такі фактори як:

1. **Автоматизація рутинних завдань**: Машинне навчання дозволяє автоматизувати процеси, що вимагають значних людських ресурсів.

2. **Покращення точності**: Алгоритми машинного навчання можуть обробляти величезні обсяги даних та знаходити закономірності, які людина може не помітити, що підвищує точність прогнозів та рішень.
3. **Адаптивність**: Моделі машинного навчання можуть адаптуватися до нових даних, покращуючи свої прогнози з часом.
4. **Ефективне використання ресурсів**: Автоматизація аналізу даних та прийняття рішень дозволяє ефективніше використовувати людські та матеріальні ресурси.

В свою чергу недоліками можна назвати:

1. **Залежність від якості даних**: Ефективність моделей сильно залежить від якості та кількості даних, які використовуються для їх навчання. Низька якість даних може призвести до неточних або навіть шкідливих результатів.
2. **Чорний ящик**: Багато алгоритмів машинного навчання, особливо глибокі нейронні мережі, діють як «чорні ящики», тобто важко зрозуміти, як саме модель прийшла до певного рішення.
3. **Етичні питання**: Використання машинного навчання може призводити до етичних дилем, включаючи проблеми конфіденційності, упередженості алгоритмів та відповідальності за автоматизовані рішення.
4. **Висока вартість**: Розробка та впровадження моделей машинного навчання може бути дорогою та вимагати значних обчислювальних ресурсів.

2.4. Машинне навчання станом на сьогодні

На сьогоднішній день існує широкий спектр ML-алгоритмів, які можна класифікувати за різними ознаками. Залежно від способу навчання, ML-алгоритми можна розділити на надзорне навчання, безнадзорне навчання та підсилювальне навчання.

- Навчання з вчителем передбачає наявність навчальних даних, які містять як значення ознак, так і бажані значення цільової функції. Алгоритми надзорного навчання навчаються на цих даних, щоб прогнозувати значення цільової функції для нових наборів даних.
- Навчання без вчителя не передбачає наявність навчальних даних з бажаними значеннями цільової функції. Алгоритми безнадзорного навчання навчаються на наборах даних, щоб виявляти приховані закономірності в даних.
- Підсилювальне навчання передбачає, що алгоритм отримує винагороду або покарання за свої дії. Алгоритми підсилювального навчання навчаються, щоб приймати рішення, які максимізують загальну винагороду.

Залежно від типу задачі, яку необхідно вирішити, можна використовувати різні ML-алгоритми. Наприклад, для задач класифікації часто використовуються алгоритми логістичної регресії, децезорів дерева рішень та нейронних мереж. Для задач регресії часто використовуються алгоритми лінійної регресії, поліноміальної регресії та нейронних мереж. Для задач кластеризації часто використовуються алгоритми кластеризації k-середніх, кластеризації k-медоїд та кластеризації головних компонент.

2.5. Контекстні моделі

Контекстні моделі набули значного поширення у різних галузях науки та техніки, особливо в інформаційних технологіях та штучному інтелекті. Вони дозволяють враховувати контекст при обробці інформації, що значно підвищує точність та ефективність алгоритмів.

Контекстні моделі являють собою математичні або концептуальні конструкції, які враховують зовнішні фактори та умови, що впливають на певний процес або систему. Вони відображають зв’язок між різними змінними, враховуючи їхній

взаємозв'язок і вплив середовища. Умовно їх можна поділити на статичні та динамічні, залежно від того, чи змінюється контекст у часі.

Контекст може включати різні аспекти, такі як:

1. **Темпоральний контекст**: Включає часові аспекти, як-от поточний час, сезон, або часовий інтервал між подіями.
2. **Просторовий контекст**: Враховує географічне розташування або просторові координати.
3. **Ситуаційний контекст**: Описує обставини або події, що відбуваються на даний момент.
4. **Особистісний контекст**: Включає інформацію про користувача, таку як їхні вподобання, історія взаємодії, фізіологічний стан тощо.

2.6. Використання контекстних моделей

Контекстні моделі знаходять широке застосування в різних сферах:

1. **Обробка природної мови (NLP)**: Використання контекстних моделей для розуміння і генерації тексту. Наприклад, моделі на основі трансформерів, такі як BERT (Bidirectional Encoder Representations from Transformers) та GPT (Generative Pre-trained Transformer), здатні враховувати контекст слів у реченнях, що дозволяє їм генерувати більш природний та зрозумілий текст.
2. **Рекомендаційні системи**: У таких системах контекстні моделі використовуються для покращення рекомендацій користувачам, враховуючи їхні попередні дії, час доби, місце знаходження та інші фактори. Наприклад, Netflix та Amazon використовують такі моделі для підвищення точності рекомендацій фільмів та товарів відповідно.
3. **Інтернет речей (IoT)**: У системах IoT контекстні моделі допомагають оптимізувати роботу пристроїв, враховуючи умови навколишнього

середовища. Наприклад, розумні термостати можуть адаптувати температуру у приміщенні залежно від часу доби та присутності людей.

4. **Мобільні додатки**: Використання контекстних моделей для персоналізації та покращення взаємодії з користувачем. Наприклад, Google Assistant використовує контекстні дані для надання більш точних відповідей та рекомендацій.
5. **Системи підтримки прийняття рішень**: У медичних системах контекстні моделі можуть використовуватися для врахування історії хвороби пацієнта, поточних симптомів та інших релевантних факторів для надання більш точних діагнозів і рекомендацій.
6. **Кібербезпека**: Використання контекстних моделей для виявлення аномалій і загроз в інформаційних системах, враховуючи поведінку користувачів та мережевий трафік.

Розглянемо кілька реальних прикладів застосування контекстних моделей:

1. **BERT у пошукових системах**: Google інтегрував модель BERT у свій пошуковий механізм для поліпшення розуміння запитів користувачів. Завдяки цьому пошукова система краще розуміє контекст запиту, що дозволяє надавати більш релевантні результати.
2. **Рекомендаційна система Amazon**: Amazon використовує контекстні моделі для аналізу поведінки користувачів на сайті. Враховуючи контекст, система пропонує товари, які можуть зацікавити конкретного користувача, підвищуючи таким чином ймовірність покупки.
3. **Контекстні чат-боти**: Наприклад, чат-боти у банківських системах використовують контекстні моделі для кращого розуміння запитів клієнтів. Це дозволяє надавати більш точні та корисні відповіді, враховуючи попередні запити та історію взаємодії.

4. **Медичні системи підтримки рішень**: У сфері охорони здоров'я контекстні моделі допомагають лікарям приймати більш обґрунтовані рішення, враховуючи детальну історію хвороби пацієнта, його поточний стан, а також результати попередніх аналізів і обстежень.
5. **Розумні будинки**: Використання контекстних моделей у системах розумного будинку дозволяє автоматизувати різні процеси, такі як освітлення, опалення та безпека, враховуючи поточні умови та поведінку мешканців.
6. **Маркетинг та реклама**: У сфері маркетингу контекстні моделі дозволяють краще розуміти потреби та вподобання клієнтів, забезпечуючи більш таргетовану та ефективну рекламу.

2.7. Переваги та недоліки контекстних моделей

Перевагами контекстних моделей є:

- **Підвищена точність**: Врахування контексту дозволяє більш точно інтерпретувати дані та приймати рішення.
- **Персоналізація**: Контекстні моделі сприяють кращій персоналізації послуг і продуктів для користувачів, що підвищує їх задоволеність.
- **Адаптивність**: Такі моделі можуть адаптуватися до змін умов і середовища, що робить їх більш гнучкими та ефективними у використанні.
- **Покращена взаємодія з користувачем**: Завдяки контекстним моделям системи можуть краще розуміти та реагувати на потреби користувачів, що підвищує якість їхнього досвіду.

В свою чергу, недоліками є:

- **Складність розробки**: Створення контекстних моделей вимагає значних зусиль і ресурсів, включаючи збір та обробку великої кількості даних.

- **Вимоги до обчислювальних ресурсів**: Для обробки контекстної інформації потрібні потужні обчислювальні ресурси, що може бути дорогим.
- **Конфіденційність та безпека**: Використання контекстних даних може підняти питання конфіденційності та безпеки, особливо якщо мова йде про персональні дані користувачів.
- **Проблеми із збиранням даних**: Для побудови ефективних контекстних моделей потрібні точні та релевантні дані, збір яких може бути складним і часом затратним процесом.
- **Складність інтеграції**: Інтеграція контекстних моделей у існуючі системи може вимагати значних змін та налаштувань, що може бути складним та витратним процесом.

РОЗДІЛ 3. СЕРВІС DIALOGFLOW, OPENAI API ТА МОДЕЛІ GPT

3.1. Сервіс Dialogflow

Платформа Dialogflow використовує у собі різні методи машинного навчання, такі як навчання з вчителем, так і навчання без вчителя, тому у практичній частині курсової роботи, проведене дослідження, за допомогою даного сервісу, нижче описана основна інформація про цю платформу.

Dialogflow - це платформа розуміння природної мови, яка дозволяє легко розробляти та інтегрувати розмовну користувацьку інтерфейс (UI) у мобільні додатки, веб-додатки, пристрої, чат-боти, системи інтерактивного відгуку (IVR) тощо.

Dialogflow може аналізувати різні типи вхідних даних від користувачів, включаючи текстові або аудіо вхідні дані (наприклад, з телефону або голосового запису). Він також може відповідати користувачам двома способами: текстом або синтетичним мовленням.

Dialogflow доступний у двох редакціях: Dialogflow CX (розширена) та Dialogflow ES (стандартна).

3.2. Основні принципи роботи Dialogflow

Dialogflow працює за допомогою наступних основних принципів:

- Розуміння природної мови: Dialogflow використовує машинне навчання для розуміння природного мовного введення користувачів. Він може розпізнавати слова та фрази, а також визначати їхній контекст.
- Генерація природної мови: Dialogflow може генерувати природний мовний текст як відповідь на введення користувача. Він може використовувати цей текст для надання інформації, виконання завдань або просто для спілкування з користувачем.
- Інтеграція з іншими системами: Dialogflow можна легко інтегрувати з іншими системами, такими як мобільні додатки, веб-додатки, пристрої та системи IVR.

3.3. Архітектура сервісу Dialogflow

Архітектура сервісу Dialogflow складається з наступних основних компонентів:

- Agent: Агент – це основний компонент Dialogflow, який визначає поведінку розмовного UI. Він містить такі елементи, як семантичні моделі, діалогові сценарії та правила.
- Intent: Намір – це позначення типу дії, яку користувач хоче виконати. Наприклад, намір «Замовити продукт» означає, що користувач хоче замовити продукт.

- Entity: Сутність – це позначення реального об'єкта або поняття. Наприклад, ентність «Продукт» може представляти будь-який продукт, який продається компанією.
- Context: Контекст – це стан розмови. Він зберігає інформацію про те, що відбулося раніше в розмові.

3.4. Функціональність сервісу Dialogflow

Dialogflow пропонує широкий спектр функцій, які дозволяють розробникам створювати розмовні UI для різних цілей. До основних функцій відносяться:

- Розуміння природної мови: Dialogflow може розуміти різні типи природного мовного введення, включаючи текст, голос і мову жестів.
- Генерація природної мови: Dialogflow може генерувати природний мовний текст як відповідь на введення користувача.
- Інтеграція з іншими системами: Dialogflow можна легко інтегрувати з іншими системами, такими як мобільні додатки, веб-додатки, пристрої та системи IVR.
- Адміністрування та моніторинг: Dialogflow пропонує інструменти для адміністрування та моніторингу розмовних UI.

3.5. Додаткові можливості сервісу Dialogflow

Крім основних функцій, Dialogflow також пропонує ряд додаткових можливостей, які можуть бути корисними для розробників. До цих можливостей відносяться:

- Підтримка мультимовності: Dialogflow підтримує понад 100 мов.
- Підтримка розшифровки мовлення: Dialogflow може розшифровувати мовлення в реальному часі.

- Підтримка штучного інтелекту: Dialogflow можна використовувати для створення розмовних UI, які використовують штучний інтелект.

3.6. Переваги використання Dialogflow

Dialogflow має ряд переваг, що включають:

- Легкість використання. Dialogflow має простий і інтуїтивно зрозумілий інтерфейс, який дозволяє розробникам легко створювати розмовні інтерфейси.
- Dialogflow пропонує широкий спектр інструментів та функцій, які спрощують процес створення розмовних інтерфейсів. Наприклад, Dialogflow має вбудований редактор діалогів, який дозволяє розробникам легко створювати сценарії діалогів. Dialogflow також має вбудований редактор відповідей, який дозволяє розробникам легко генерувати відповіді на запити користувачів.
- Гнучкість. Dialogflow можна використовувати для створення широкого спектру розмовних інтерфейсів, від простих ботів до складних IVR-систем.
- Dialogflow є досить гнучкою платформою, яка може бути використана для створення різних типів розмовних інтерфейсів. Наприклад, Dialogflow можна використовувати для створення чат-ботів, голосових помічників, систем IVR тощо.
- Можливість масштабування. Dialogflow можна легко масштабувати відповідно до зростаючих потреб бізнесу.
- Dialogflow є масштабованою платформою, яка може підтримувати великі обсяги трафіку. Наприклад, Dialogflow можна використовувати для обслуговування великої кількості клієнтів або для обробки великих обсягів даних.

3.7. Приклади використання Dialogflow

Dialogflow можна використовувати для створення різних розмовних інтерфейсів, включаючи:

- Чат-боти. Чат-боти — це програми, які можуть спілкуватися з користувачами за допомогою природної мови. Dialogflow можна використовувати для створення чат-ботів, які можуть надавати інформацію, відповідати на запитання або виконувати завдання. Наприклад, Dialogflow можна використовувати для створення чат-бота для служби підтримки клієнтів, який може допомагати клієнтам вирішувати проблеми.
- Голосові помічники. Голосові помічники — це програми, які можуть розуміти і реагувати на голосові команди. Dialogflow можна використовувати для створення голосових помічників, які можуть виконувати різні завдання, наприклад, відтворювати музику, надавати інформацію або керувати пристроями. Наприклад, Dialogflow можна використовувати для створення голосового помічника, який може допомагати користувачам керувати своїми розумними будинками.
- Системи IVR. Системи IVR (інтерактивне голосове реагування) — це системи, які використовують голосові команди для надання послуг клієнтам. Dialogflow можна використовувати для створення систем IVR, які можуть надавати інформацію, відповідати на запитання або виконувати завдання. Наприклад, Dialogflow можна використовувати для створення системи IVR для служби підтримки клієнтів, яка може допомагати клієнтам бронювати рейси.
- Пристрої. Dialogflow можна використовувати для створення розмовних інтерфейсів для пристроїв, таких як смартфони, планшети, розумні колонки тощо. Наприклад, Dialogflow можна використовувати для створення

розмовного інтерфейсу для розумної колонки, яка може надавати інформацію, відповідати на запитання або виконувати завдання.

Ось кілька конкретних прикладів використання Dialogflow:

- Google Assistant. Google Assistant — це голосовий помічник, який використовує Dialogflow для розуміння і реагування на голосові команди.
- Amazon Alexa. Amazon Alexa ─ це ще один голосовий помічник, який використовує Dialogflow.
- Cisco Webex Assistant. Cisco Webex Assistant — це система IVR, яка використовує Dialogflow для надання послуг клієнтам.
- Tata Consultancy Services (TCS). TCS використовує Dialogflow для створення чат-бота, який допомагає клієнтам вирішувати проблеми з їхніми продуктами і послугами.
- Kroger. Kroger використовує Dialogflow для створення чат-бота, який допомагає клієнтам економити час і гроші при покупках.

3.8. Сервіс OpenAI API

OpenAI API — це інтерфейс прикладного програмування (API), розроблений компанією OpenAI, що надає можливість інтегрувати передові моделі штучного інтелекту (ШІ) в різноманітні програми та сервіси. Використовуючи цей API, розробники можуть додавати функціональність, пов'язану з обробкою природної мови, до своїх застосунків, що дозволяє здійснювати широкий спектр завдань від автоматичного створення тексту до обробки запитань і відповідей, аналізу настроїв та інше.

3.9. Основні принципи роботи OpenAI API

OpenAI API працює за принципом відправлення запитів до серверів OpenAI, де обробляються завдання за допомогою попередньо навчених моделей ШІ. Ключові елементи роботи з API включають:

1. **Запити і відповіді**: Розробник відправляє запит (request) з даними до сервера OpenAI через HTTP. Сервер обробляє запит за допомогою моделей ШІ і повертає відповідь (response) у форматі JSON.
2. **Токени**: Основна одиниця вимірювання роботи з текстом у OpenAI API — токени, які можуть бути словами або частинами слів. Вартість використання API залежить від кількості використаних токенів.
3. **Ключ API**: Для доступу до API потрібен унікальний ключ API, який отримується після реєстрації на платформі OpenAI. Цей ключ використовується для автентифікації запитів.

3.10. Архітектура OpenAI API

Архітектура OpenAI API складається з кількох основних компонентів:

1. **Клієнтська частина**: Будь-яка програма або сервіс, який відправляє запити до API.
2. **Мережевий інтерфейс**: Канал зв'язку, що забезпечує передачу даних між клієнтом і серверами OpenAI.
3. **Серверна частина**: Сервери OpenAI, що містять обчислювальні ресурси та моделі ШІ, які обробляють запити.
4. **Моделі ШІ**: Високопродуктивні моделі на основі архітектури трансформерів, наприклад, GPT-3 і GPT-4, які здійснюють аналіз та генерацію тексту.

3.11. Функціональність OpenAI API

OpenAI API надає широкий спектр функцій, які можна використовувати у різноманітних сценаріях:

1. **Генерація тексту**: Створення текстів на основі заданого початкового контексту.
2. **Аналіз тексту**: Визначення сенсу, настрою, тематики тексту.
3. **Переклад**: Автоматичний переклад текстів між різними мовами.
4. **Запитання і відповіді**: Відповіді на запитання на основі наявної інформації.
5. **Пошук та витяг інформації**: Знаходження релевантної інформації у великих текстових масивах.

3.12. Додаткові можливості OpenAI API

Крім основних функцій, OpenAI API пропонує також додаткові можливості:

1. **Модифікація моделей**: Налаштування моделей під специфічні потреби завдяки параметрам, які можна змінювати.
2. **Фільтри змісту**: Використання фільтрів для запобігання генерації неприпустимого або шкідливого контенту.
3. **Інтеграція з іншими сервісами**: Можливість інтеграції API з іншими веб-сервісами та платформами через веб-хуки та інші механізми.
4. **Обробка великих обсягів даних**: Масштабованість API дозволяє працювати з великими обсягами даних, зберігаючи високу продуктивність.

3.13. Переваги та недоліки використання OpenAI API

Переваги:

1. **Висока точність**: Моделі OpenAI забезпечують високу точність та якість генерації тексту.

2. **Широка функціональність**: Великий набір функцій для різних завдань обробки тексту.
3. **Гнучкість**: Можливість налаштування моделей під специфічні потреби.
4. **Масштабованість**: API може обробляти великий обсяг запитів одночасно.
5. **Легкість інтеграції**: Простий у використанні інтерфейс та добре документована API.

Недоліки:

1. **Вартість**: Використання OpenAI API може бути дорогим для великих проєктів або стартапів.
2. **Залежність від Інтернету**: Потребує постійного доступу до Інтернету для роботи.
3. **Обмеження на використання**: Існують обмеження на використання API для певних типів контенту через політику OpenAI.
4. **Конфіденційність даних**: Передача даних до зовнішнього сервера може бути проблематичною з точки зору конфіденційності.

3.14. Приклади використання OpenAI API

Такі приклади використання OpenAI API можна назвати:

1. **Автоматизація підтримки клієнтів**: Використання API для створення чат-ботів, які можуть відповідати на запити клієнтів у реальному часі, зменшуючи навантаження на службу підтримки.
2. **Контент-маркетинг**: Генерація статей, постів у блогах та соціальних мережах, що значно знижує час та зусилля на створення контенту.
3. **Освітні платформи**: Використання OpenAI для розробки інтерактивних освітніх ресурсів, що допомагають студентам у навчанні, забезпечуючи відповіді на запитання та пояснення складних концепцій.
4. **Персоналізовані рекомендації**: Аналіз поведінки користувачів та надання персоналізованих рекомендацій у сфері електронної комерції, медіа та розваг.

5. **Наукові дослідження**: Використання API для аналізу великих обсягів наукових текстів та пошуку релевантної інформації для дослідницьких проєктів.

3.15. Огляд моделей GPT

Моделі GPT (Generative Pre-trained Transformer) є основою технологій OpenAI API. Ці моделі представляють собою серію покращених версій алгоритмів машинного навчання, які демонструють вражаючу здатність до обробки природної мови (Natural Language Processing, NLP). У цьому розділі ми розглянемо еволюцію моделей GPT, їх основні характеристики та наукові підходи, що лежать в їх основі.

Далі будуть розглядатися моделі від **GPT-1** до **GPT-4.**

1. **GPT-1**
   - **Випуск**: 2018 рік.
   - **Кількість параметрів**: 117 мільйонів.
   - **Архітектура**: Першу версію моделі було побудовано на основі трансформерної архітектури, запропонованої Васвані та співавторами у 2017 році. GPT-1 навчалась на великих обсягах текстових даних з метою предиктивного моделювання слів у контексті.
   - **Особливості**: Вперше продемонстрована ефективність підходу переднавчання (pre-training) з подальшим тонким налаштуванням (fine-tuning) для виконання специфічних завдань NLP.
2. **GPT-2**
   - **Випуск**: 2019 рік.
   - **Кількість параметрів**: 1.5 мільярда.
   - **Архітектура**: Модель GPT-2 значно перевершує свого попередника завдяки збільшенню кількості параметрів та обсягу навчальних даних. Вона продовжує використовувати трансформерну архітектуру.

- **Особливості**: GPT-2 здатна генерувати зв'язні тексти великого обсягу, демонструючи значно кращі результати в різних задачах NLP. Розробники підняли питання етичного використання моделі через її потенціал до створення дезінформації.

3. **GPT-3**
   - **Випуск**: 2020 рік.
   - **Кількість параметрів**: 175 мільярдів.
   - **Архітектура**: GPT-3 є значним проривом у розвитку трансформерних моделей завдяки величезному збільшенню кількості параметрів. Вона навчається на ще більших обсягах даних і демонструє високий рівень загальної мовної компетенції.
   - **Особливості**: Модель здатна виконувати широкий спектр завдань без спеціального тонкого налаштування. Вона розуміє контекст краще, ніж попередні версії, і може генерувати тексти, що мало відрізняються від тих, які пишуть люди.
4. **GPT-4**
   - **Випуск**: 2023 рік.
   - **Кількість параметрів**: Точна кількість параметрів не розголошується, але відомо, що GPT-4 включає ще більше параметрів та покращену архітектуру порівняно з GPT-3.
   - **Архітектура**: GPT-4 базується на новітніх наукових досягненнях у сфері трансформерних моделей і включає покращення в обробці контексту, ефективності навчання та здатності до генерації тексту.
   - **Особливості**: Модель демонструє високий рівень розуміння природної мови, здатність до вирішення складних завдань та підтримку багатомовних застосувань. Вона також включає вдосконалені механізми безпеки для запобігання зловживанням.

3.16. Наукові основи моделей GPT

Моделі GPT базуються на концепції трансформерів, які були запропоновані в роботі «Attention is All You Need» (Васвані та ін., 2017). Основні принципи, які лежать в основі моделей GPT, включають:

1. **Механізм самоуваги (Self-Attention)**: Трансформери використовують механізм самоуваги, що дозволяє моделі звертати увагу на різні частини вхідного тексту для побудови зв'язків між словами незалежно від їх відстані у реченні.
2. **Переднавчання на великих корпусах тексту**: Моделі GPT навчаються на величезних обсягах неструктурованих текстових даних, що дозволяє їм формувати загальні знання про мову і контекст.
3. **Підхід «без учителя» (Unsupervised Learning)**: Навчання моделей здійснюється без використання спеціально розмічених даних, що робить їх гнучкими для різноманітних задач NLP.
4. **Тонке налаштування (Fine-Tuning)**: Хоча моделі можуть ефективно працювати без додаткового налаштування, для певних задач можна виконати тонке налаштування на специфічних наборах даних для покращення продуктивності.
5. **Масштабування моделей**: Значне збільшення кількості параметрів моделей GPT дозволяє досягати кращих результатів у порівнянні з попередніми версіями, демонструючи переваги підходу масштабування в машинному навчанні.

РОЗДІЛ 4. РОЗРОБКА ЧАТ-БОТУ

4.1. Реєстрація боту

BotFather – найпростіший та доступний спосіб для реєстрації, налаштування та управління своїм телеграм ботом. Робота з ним проста і не вимагає професіональних навичок. За допомогою BotFather можна зареєструвати необмежену кількість нових чат-ботів. Єдиною умовою для реєстрації нового бота — є його унікальний username. По суті, Telegram Bots — це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Користувачі можуть взаємодіяти з ботами двома способами:

- посилання повідомлення та команди ботам, відкриваючи з ними чат або додаючи їх до груп;
- посилання запитів безпосередньо з поля введення, ввівши @username бота та запит. Це дозволяє надсилати вміст від вбудованих ботів безпосередньо в будь-який чат, групу чи канал.

Повідомлення, команди та запити, надіслані користувачами, передаються програмному забезпеченню, запущеному на ваших серверах. Наш сервер-посередник обробляє за вас усе шифрування та спілкування з Telegram API. Ви спілкуєтеся з цим сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram. Ми називаємо цей інтерфейс нашим API бота.

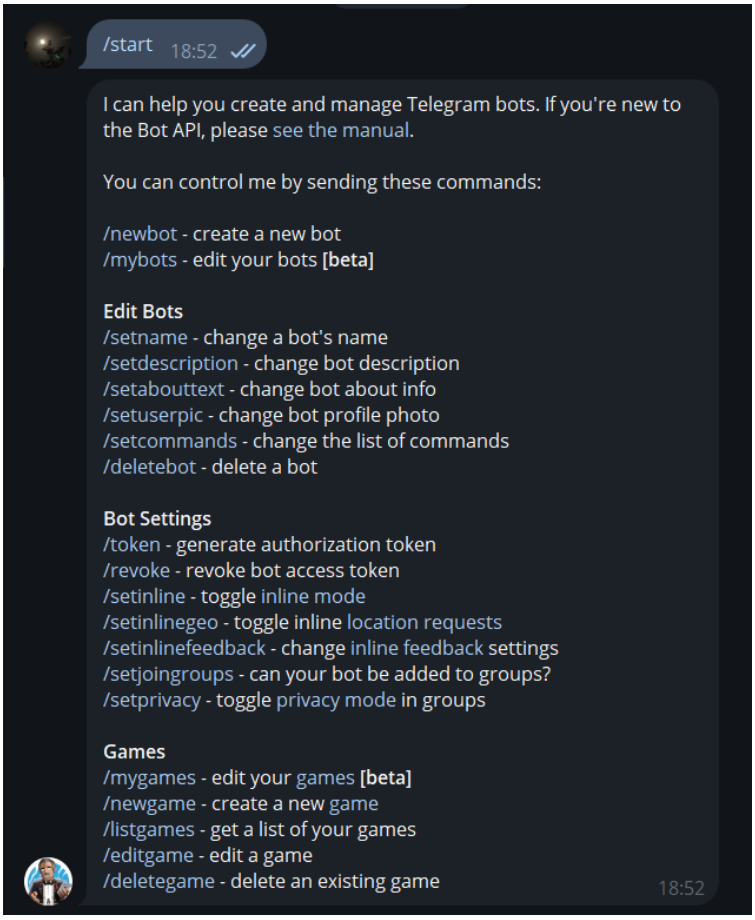

Рисунок 4.1. Команди чат-бота «BotFather»

BotFather стане відмінним рішенням для тих людей, які не розбирається в програмуванні і не хоче довіряти створення і управління своїм ботом стороннім людям або стороннім організаціям.

Як зареєструвати нового бота за допомогою BotFather? Взаємодія з BotFather здійснюється за допомогою простих команд. Наприклад, для того, щоб зареєструвати нового бота, досить відправити в чат команду */newbot* і слідувати простим інструкціям:

1. Придумати ім'я бота, яке буде відображатися в чатах і контактах. Надалі його можна буде змінити. Тут все залежить тільки від вашої фантазії і вимог;

2. Придумати username - це вже складніше: ім'я повинно бути унікальним і закінчуватися на «bot». Допускаються букви латинського алфавіту, цифри і символ підкреслення (приклад - «TestCrazy_bot»). Загальна кількість символів не менше 5 і

не більше 32;

3. Якщо все в порядку, то у відповідь ми отримаємо повідомлення з токеном. Токен необхідний для роботи з Bot API за допомогою http-протоколу. Не можна передавати його іншим і бажано не втрачати. Хорошим рішенням буде скопіювати його: зберегти в текстовий файл і покласти в надійне, завжди доступне місце - наприклад, хмарне сховище. Після реєстрації можна приступати до облагороджування бота: встановити аватар, додати опис та інше.

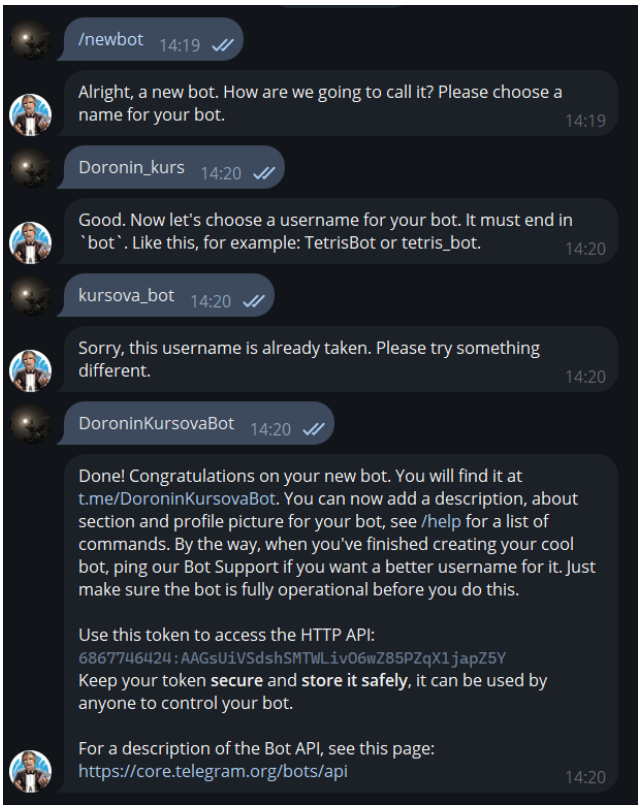

Рисунок 4.2. Отримання повідомлення з токеном

Що вміє BotFather? Крім реєстрації бота, за допомогою BotFather можна здійснювати його налаштування і управління. Якщо Вам не подобається ім`я бота, то його можна змінити командою */setname*.

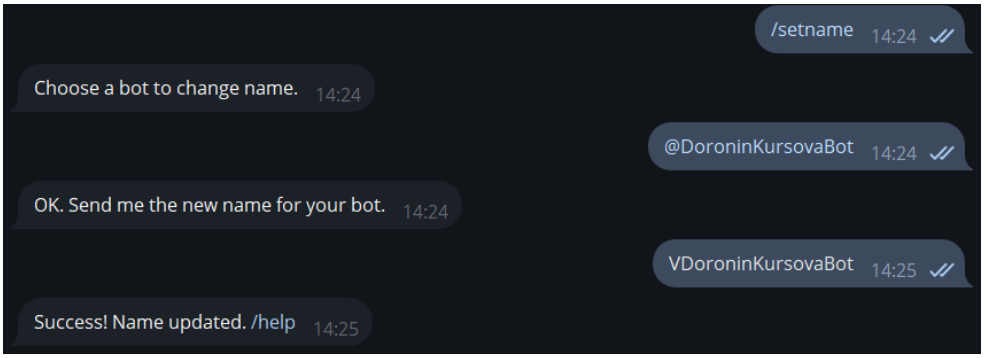

Рисунок 4.3. Зміна імені бота

Встановити фотографію профілю можна командою */setuserpic*, а змінити або додати короткий опис командою */setdescription*. Зміна інформації в профілі виконується командою */setabouttext*. На цьому розробка зовнішнього вигляду бота можна вважати закінченою, і можна зайнятися його функціональними можливостями.

Функціонал Картинка і опис – це, звичайно теж важливо, але все ж таки, головне в боті - це його можливості. Можливості залежать від набору команд, які здатний обробити бот. Для встановлення списку команд введіть */setcommands* у вікні чату.

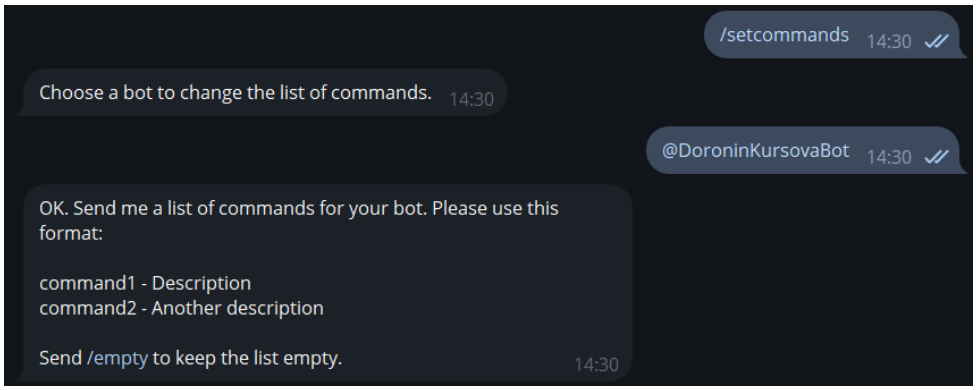

Рисунок 4.4. Встановлення команд для бота

Команди вводяться з / в форматі «/*command1* - опис команди» Ось приклади написання основних команд і їх розшифровка:

- /*Newbot* – зареєструвати нового бота;
- /*Mybots* – редагувати своїх ботів [бета];
- /*Setname* – змінити ім'я бота;
- /*Setdescription* – змінити опис бота;
- /*Setabouttext* – змінити інформацію про боті;
- /*Setuserpic* – змінити фотографію профілю бота;
- /*Setcommands* – змінити список команд;
- /*Deletebot* – видалити бота. Налаштування бота;
- /*Token* – генерувати токен авторизації;
- /*Revoke* – відкликати токен доступу до боту;
- /*Setinline* – включити вбудований режим (дозволяє звертатися безпосередньо до боту з будь-якого каналу, групи або чату, написавши його ім'я в поле відправки повідомлень);
- */Setinlinegeo* – перемикати запити про розташування при використанні бота у вбудованому режимі.

4.2. Платформа Dialogflow та OpenAI API

Як і вказувалось у розділі вище, Dialogflow – це хмарний сервіс розпізнавання природної мови від Google, який підтримує різні мови, зокрема російську. Він має безкоштовні ліміти використання, а для роботи з API можна скористатися бібліотеками для різних мов, тому його досить легко додати до своїх проектів. Також у Dialogflow «з коробки» є взаємодія з різними месенджерами, так що для простих сценаріїв написання свого коду може навіть не знадобитися. Спочатку потрібно створити технічного спеціліста, який виконуватиме основну роботу зі спілкування з користувачем. Для розуміння суті спеціаліста DF можна провести паралель зі співробітником call- центру, який обробляє запити клієнта

(користувача).

Потрібно придумати його назву, вибрати основну мову спілкування та часовий пояс, в якому він буде працювати.

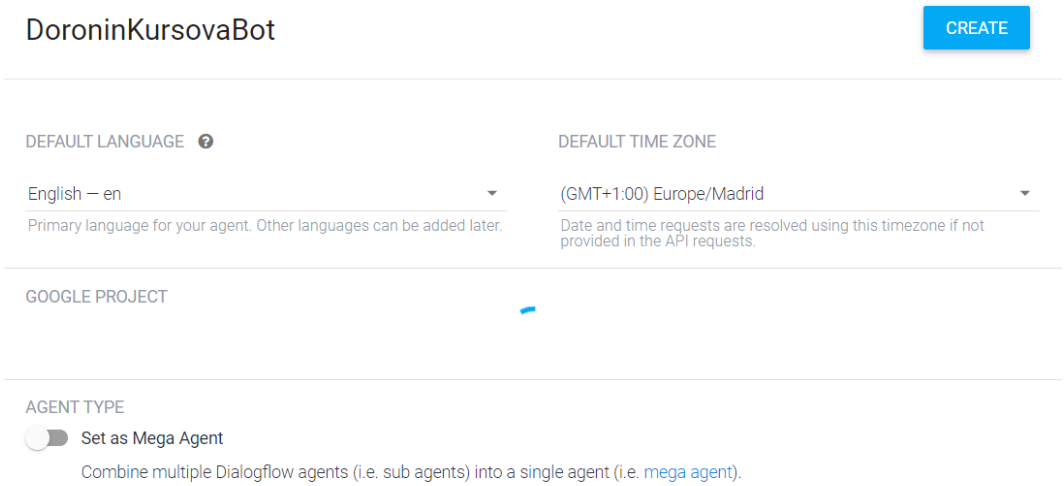

Рисунок 4.5. Створення персонального агенту

Коли наш технічний спеціаліст був створенний, потрібно його зв’язати з нашим телеграм чат-ботом:

Рисунок 4.6. Інтегрування Dialogflow з телеграм-ботом

Щоб агент почав обробляти запити користувача, потрібно додати до нього Intents (наміри, цілі). Можна сказати, що вони повинні відповідати намірам користувача, який спілкується з чат-ботом. Наприклад, купити щось, отримати

якусь інформацію, тощо.

Як правило, після створення спеціаліста, в ньому вже присутні одразу дві мети: одна – для реакції на вітання та початок діалогу (Default Welcome Intent), та інша – спеціальна, на той випадок, якщо не вдалося нічого розпізнати (Default Fallback Intent). У кожному з намірів можна налаштувати «ознаки», за якими відбуватиметься перехід саме до нього. Найпростіше - це додати тренувальні фрази (Training phrases), на основі яких Dialogflow визначає той чи інший намір користувача.

Також можна вказувати події (Events), використовуючи стандартні або придумавши щось своє. Тоді перехід у Intent можна буде форсувати, передавши у запиті до DF потрібну назву:

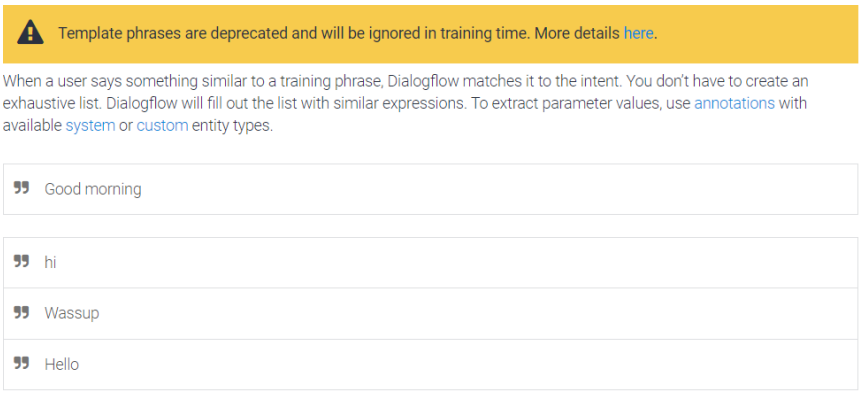

Рисунок 4.7. Можливі фрази користувача

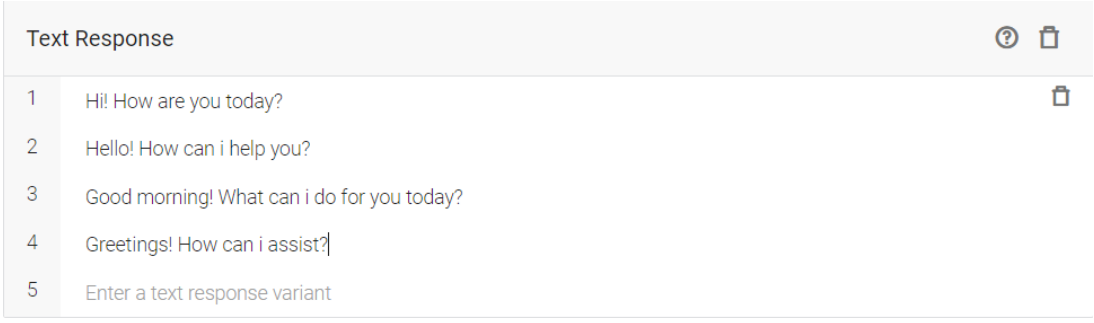

Рисунок 4.8. Можливі відповіді чат-бота

Тепер чат-бот вміє відповідати на вітання. При розробці навичок для голосових помічників часто рекомендують, щоб в одній і тій же ситуації вони не видавали однакові відповіді. Використовуючи Dialogflow, можна не турбуватися про це, тому що сервіс випадково вибирає одну з фраз, зазначених у розділі Responses. Перевірити це можна в «тестовій консолі», що знаходиться праворуч або в онлайн месенджері «Телеграм»:

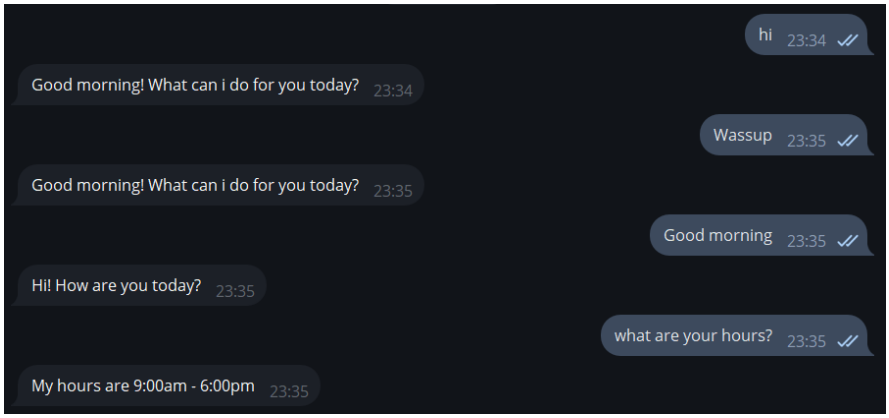

Рисунок 4.9. Тестування вітання с ботом

Однією з корисних можливостей Dialogflow можна назвати розпізнавання сутностей. Він добре справляється з більшістю спільних об'єктів: датами, містами, навіть музичними гуртами та іншим. Кожну тренувальну фразу наміру можна розмітити і таким чином вказати, який об'єкт у ній шукати і в який параметр записувати розпізнане значення:

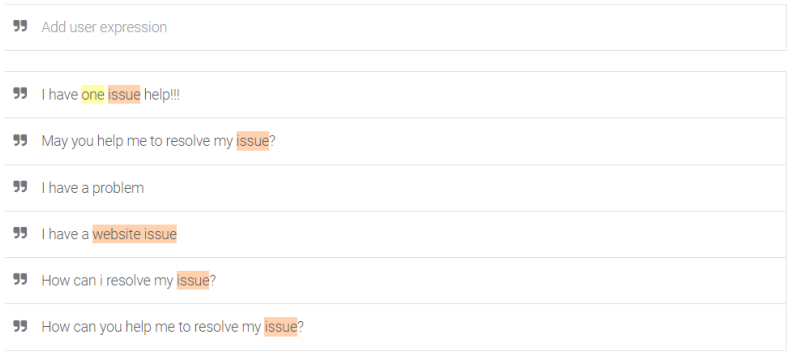

Рисунок 4.10. Створення запитів

У сервісі є безліч системних сутностей, які можуть використовуватись у більшості сценаріїв. Тут розпізнаються діапазони дат (sys.date-period), «цілісні» дати (sys.date), повні назви міст (sys.geo-city), та групи (music-artist):

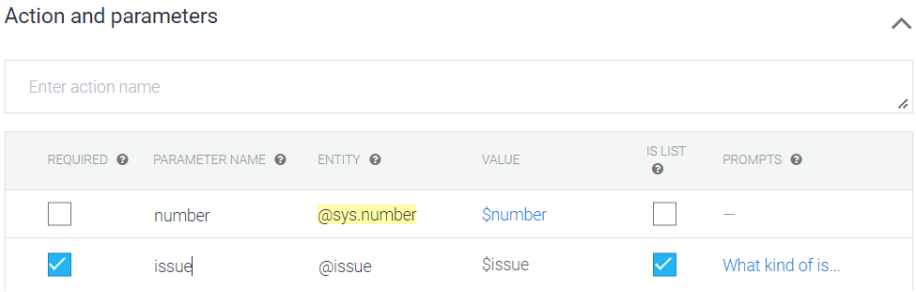

Рисунок 4.11. Параметри

Також головним етапом є створення елементів параметру «issue» за допомогою вікна «Entities», де відображено список системних об'єктів, що використовуються. На ті з них, які доступні для доповнення, можна клацнути, і в лівій колонці вказати повну назву об'єкта, а праворуч його синоніми:

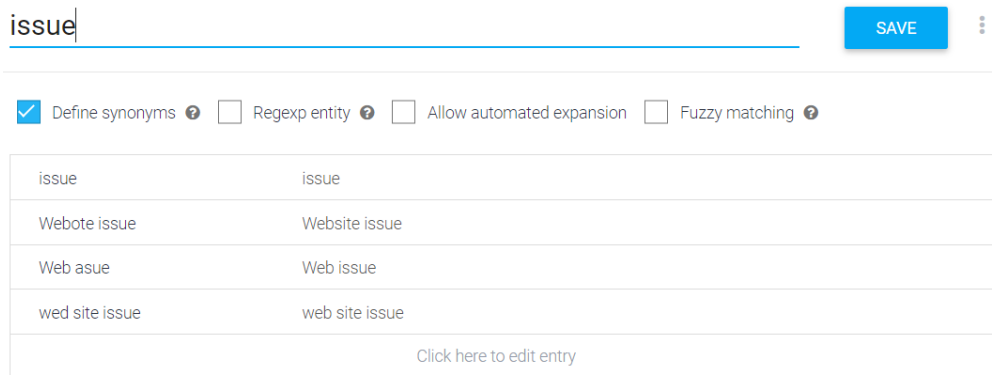

Рисунок 4.13. Елементи параметру «issue»

Якщо користувач не уточнить, яка саме проблема його турбує, то технічний спеціаліст його не зрозуміє:

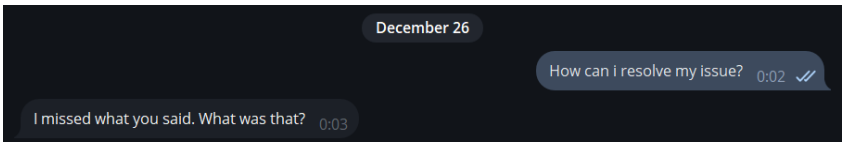

Рисунок 4.14. Тестування

Для вирішення даної проблеми, потрібно дати відсилку в параметрі «issue», на попередньо задані елементи проблеми:

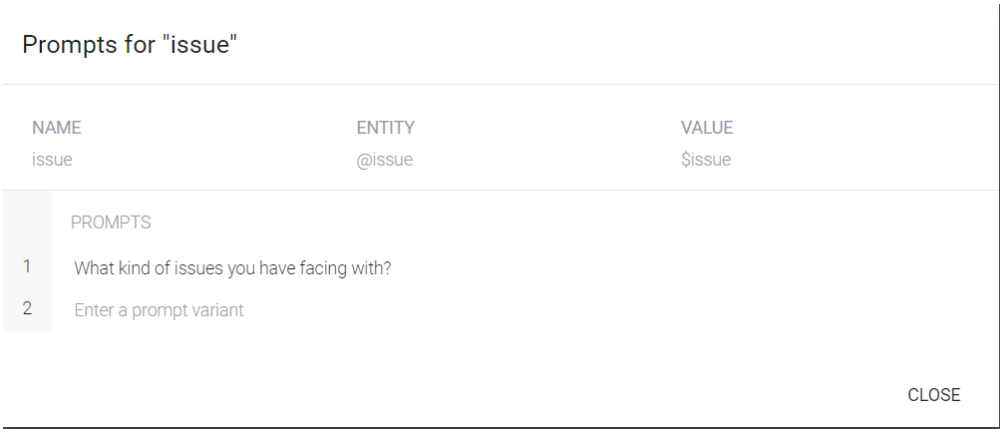

Рисунок 4.15. Параметр «issue»

Проведемо тестування щойно створеного параметру:

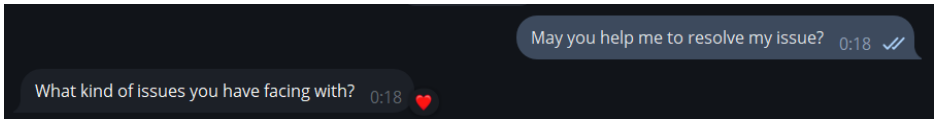

Рисунок 4.16. Тестування чат-боту

За допомогою Dialogflow можна побудувати такий сценарій, у якому розмова піде тією чи іншою «гілкою», залежно від вибору користувача. Від цього залежить, що агент запитає далі: яку начинку додати або напій.

Найпростіший спосіб зробити «гілку» - це створити пов'язаний Intent за допомогою кнопки «Add follow-up intent», яка з'являється при наведенні на той чи

інший намір. Після цього під «батьківським» елементом відображатимуться «дочірні», для яких також можна створити пов'язані наміри:

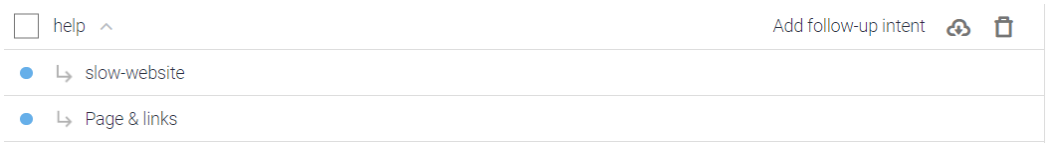

Рисунок 4.17. Створення гілки за допомогою «Add follow-up intent»

Наприклад, користувач відповів, що його проблема пов’язана з поламаною веб-сторінкою або посиланням, тоді поточний діалог має контекст «help-followup». Для відповіді на подібне запитання заведено намір «Page & links» з відповідними тренувальними фразами та відповідями, де задається наступне питання:

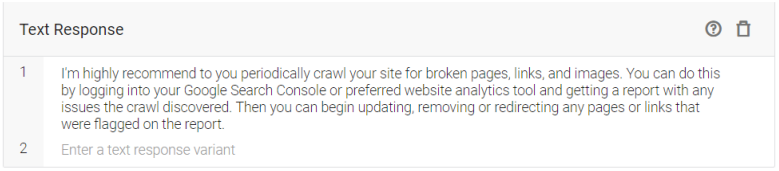

Рисунок 4.18. «Позитивна» гілка

Якщо ж користувач скаже, що проблема пов’зяна з швидкістю веб-сайта, то розмова потрапить у інший намір, в якому тренувальні фрази такі самі, але відповідь трохи інша:

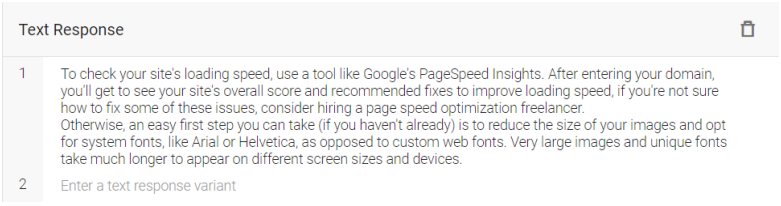

Рисунок 4.19. «Негативна» гілка

Діалог буде по тому сценарію, який вибере користувач:

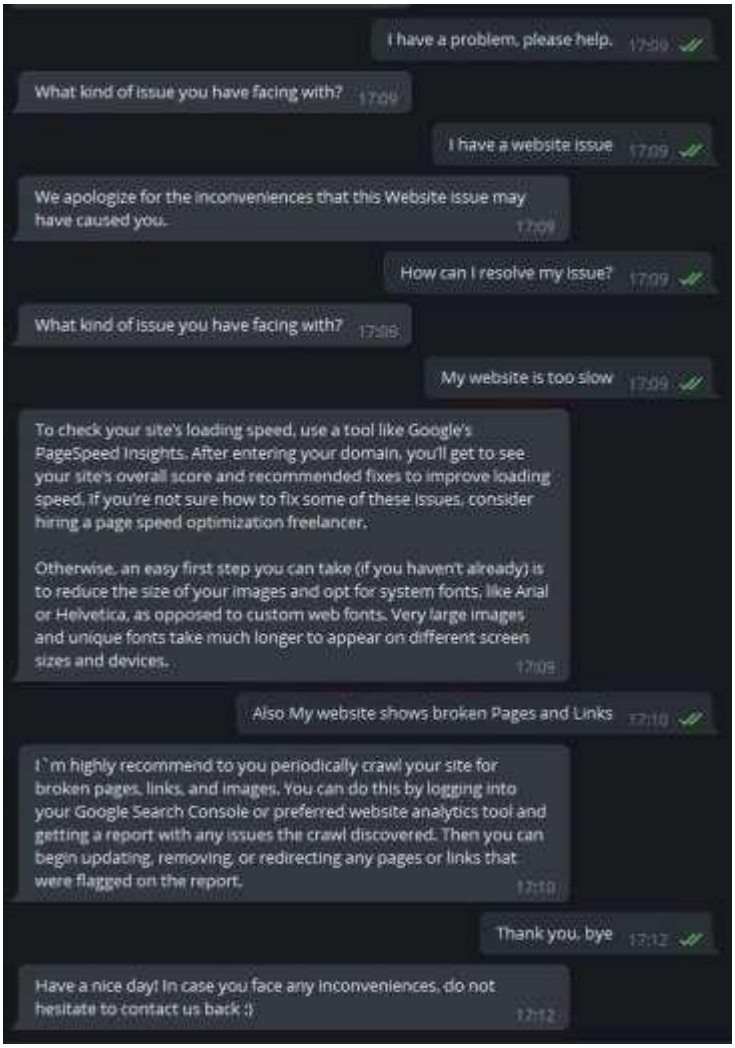

Рисунок 4.20. Тестування гілок

Також Dialogflow надає можливість інтегрувати у ваш чат-бот уже готового агента, який буде діяти згідно заданого сценарію. Ось приклади даних агентів:

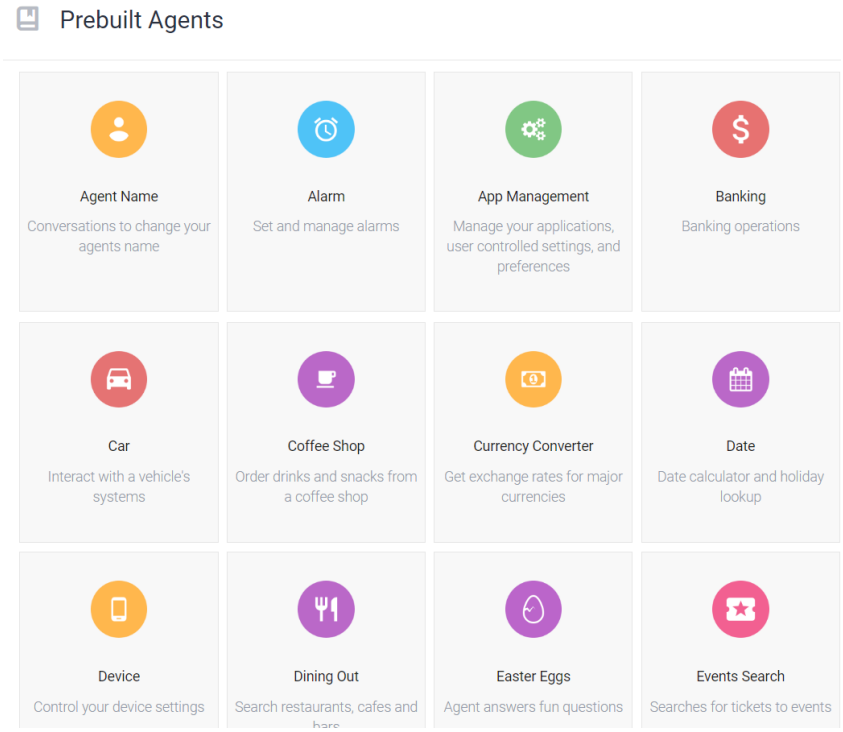

Рисунок 4.21. Готові агенти

Для того, щоб продемонструвати інтеграцію одного з агентів у чат-бот, створимо нового для прикладу на мові Python, який буде використовувати OpenAI GPT-3.5 для генерації відповідей та інтеграцію з Google Dialogflow для обробки природньої мови.

У даному дослідженні був обраний OpenAI з кількох важливих причин. По-перше, OpenAI надає широкий спектр можливостей для обробки природної мови, включаючи генерацію тексту, аналіз тексту, переклад, відповідь на запитання та пошук інформації у великих текстових масивах.

Приклади аналогів OpenAI включають Claude від Anthropic, Copilot від GitHub та Gemini. Всі ці інструменти мають свої унікальні особливості та переваги, але OpenAI виділяється за кількома параметрами:

1. **Висока точність та якість генерації тексту**: Моделі OpenAI, зокрема GPT-3 та GPT-4, демонструють високу точність та здатність генерувати тексти, що дуже схожі на ті, які пишуть люди.
2. **Широка функціональність**: OpenAI API підтримує різноманітні сценарії використання, включаючи генерацію тексту, аналіз настроїв, переклад мов та відповіді на запитання, що дозволяє використовувати його у багатьох галузях.
3. **Гнучкість та можливість налаштування**: Моделі OpenAI можна налаштовувати під специфічні потреби, що робить їх придатними для різних завдань та індустрій.
4. **Масштабованість**: OpenAI API може обробляти великий обсяг запитів одночасно, що є критично важливим для комерційних застосувань.
5. **Легкість інтеграції**: Простий у використанні інтерфейс та добре документована API роблять інтеграцію OpenAI з іншими системами швидкою та ефективною.

Щодо порівняння з аналогами:

- **Claude (Anthropic)**: Фокус на безпеку та етичність використання ШІ, але може поступатися в продуктивності та масштабованості порівняно з OpenAI.
- **Copilot (GitHub)**: Спеціалізується на асистуванні в програмуванні, використовуючи моделі OpenAI, що обмежує його універсальність у інших сферах.
- **Gemini**: Новий гравець на ринку, який ще не досяг рівня функціональності та масштабованості, як OpenAI.

Таким чином, OpenAI був обраний для цього дослідження через його високу точність, масштабованість, широку функціональність та можливість інтеграції, що робить його ідеальним вибором для розробки чат-ботів для автоматизованого навчання користувачів.

Для початку імпортуємо готового агента з назвою «Small Talk» – це агент, що уже має в собі сценарії для проведення з користувачем простих розмов, наприклад, відповісти на питання «How old are you?» і тд., для цього потрібно знайти його у списку та натиснути кнопку «IMPORT».

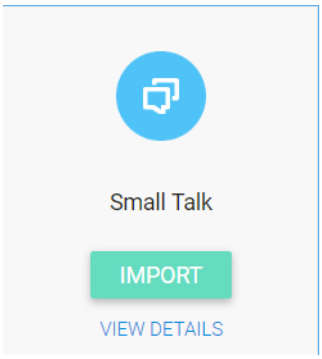

Рисунок 4.22. Імпорт агента «Small Talk»

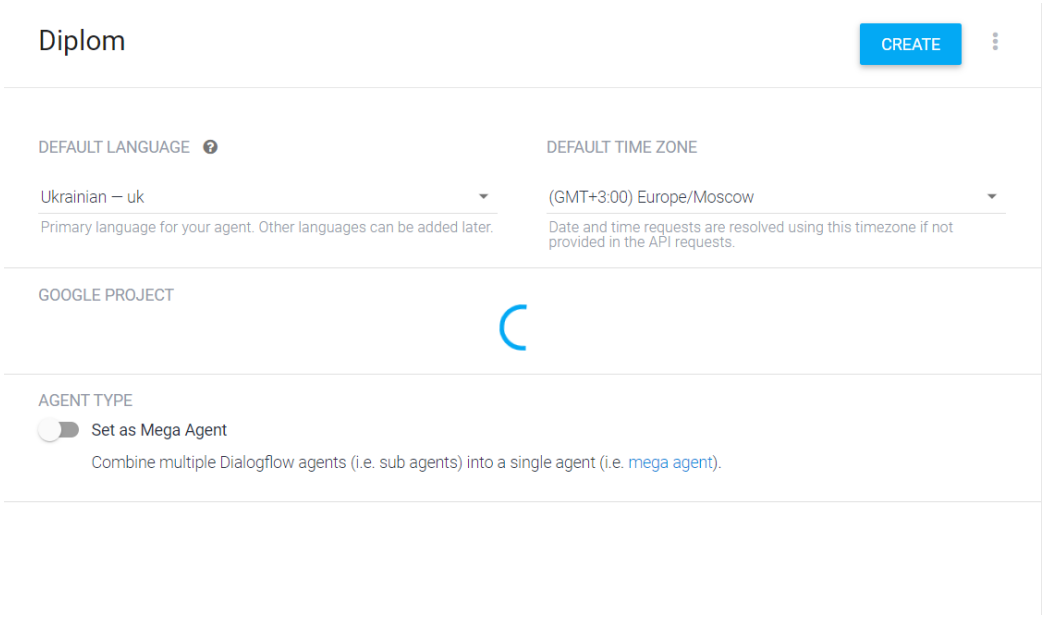

Рисунок 4.23. Налаштування агента «Small Talk»

Після цього, необхідно підготувати середовище розробки, встановивши всі необхідні бібліотеки та налаштувавши ключі доступу для використання сервісів OpenAI та Google Dialogflow. Це можна зробити за допомогою команди:

```
pip install openai google-cloud-dialogflow telegram
```

Рисунок 4.24. Команда для встановлення бібліотек

Далі імпортуємо потрібні бібліотеки у код бота.

```
import os
import openai
from google.cloud import dialogflow_v2beta1 as dialogflow
from google.api_core.exceptions import InvalidArgument
from telegram import Update
from telegram.ext import Application, ApplicationBuilder, CommandHandler, MessageHandler, filters, ContextTypes
```

Рисунок 4.25. Імпорт бібліотек

Наступним кроком буде налаштування ключа API для OpenAI, який використовується для доступу до моделі GPT-3.5. Необхідно згенерувати API-ключ на платформі OpenAI та додати його до коду.

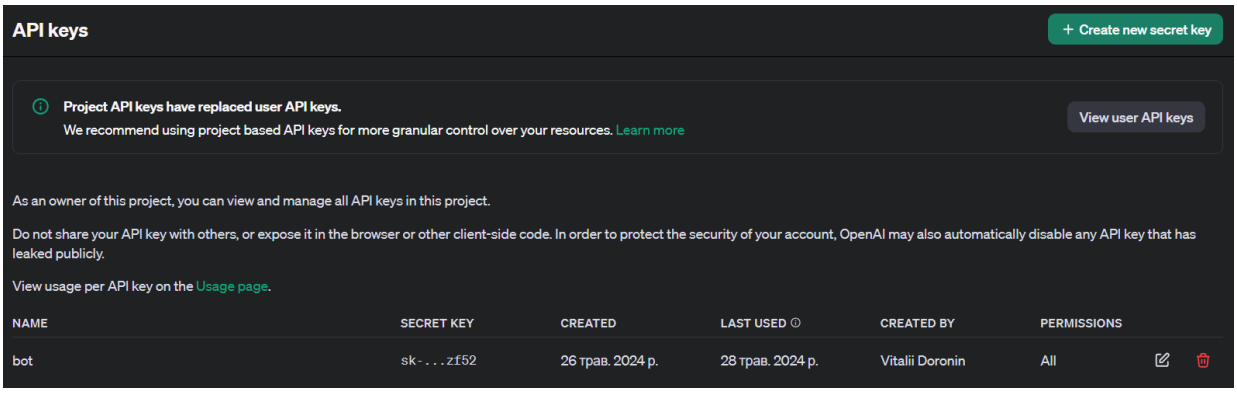

Рисунок 4.26. Створений API-ключ на платформі OpenAI

```
# Налаштування OpenAI GPT
openai.api_key = 'sk-proj-qT49hTwSgjNCeikkIc4LT3BlbkFJg5iEN1cObrTCeh7Mzf52'
```

Рисунок 4.27. API-ключ, доданий у код

Далі потрібно налаштувати ключ Google Dialogflow. Для роботи з Dialogflow необхідно створити проект у Google Cloud, згенерувати ключ JSON та вказати шлях до цього ключа у середовищі.

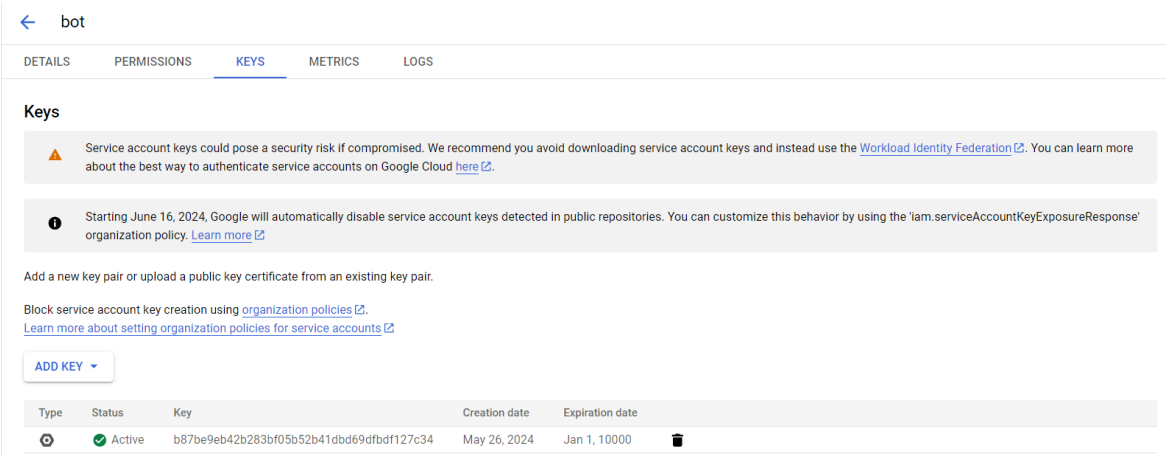

Рисунок 4.28. Створений API-ключ на платформі Google Cloud

```
# Налаштування Dialogflow
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = 'C:/Users/Admin/Desktop/диплом/small-talk1-9ftf-b87be9eb42b2.json'
DIALOGFLOW_PROJECT_ID = 'small-talk1-9ftf'
DIALOGFLOW_LANGUAGE_CODE = 'en-GB'
```

Рисунок 4.29. Налаштування ключа Google Dialogflow

Чат-бот буде взаємодіяти з користувачами через платформу Telegram. Тому для початку створимо бота через BotFather, як вже описувалось вище, і після цього необхідно буде налаштувати обробники команд та повідомлень.

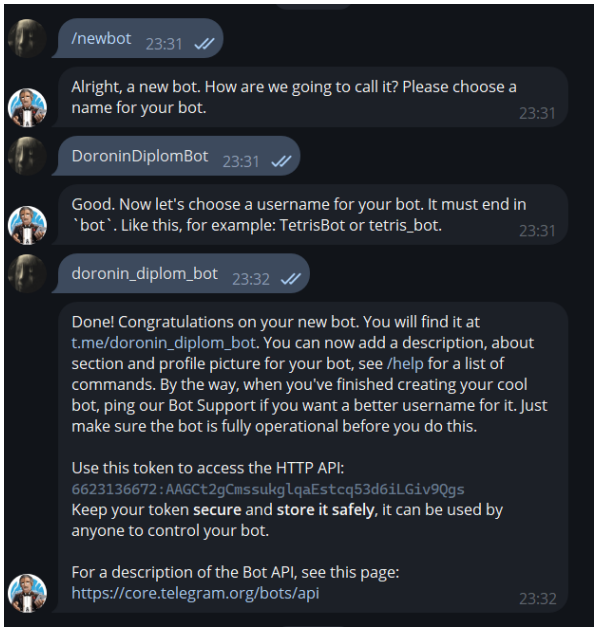

Рисунок 4.30. Створення бота через BotFather

```
application = ApplicationBuilder().token("6623136672:AAGCt2gCmssukglqaEstcq53d6iLGiv9Qgs").build()
```

Рисунок 4.31. Доданий токен чат-боту

```
application.add_handler(CommandHandler("start", start))
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
```

Рисунок 4.32. Обробники для команди */start*, та повідомлень, що не є командами

Основна логіка обробки повідомлень включає використання Google Dialogflow для первинного аналізу та OpenAI GPT-3.5 для генерації відповідей, коли Dialogflow не може надати адекватної відповіді.

Тому далі створюємо функцію обробки команд */start*, яка буде відправляти привітальне повідомлення користувачам.

```
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text("Hi! I'm your bot. How can i help you?")
```

Рисунок 4.33. Функція start

Після цього створюємо функцію «handle_message», яка буде обробляти повідомлення від користувача. Вона надсилатиме текст до Dialogflow і, якщо Dialogflow не може відповісти, використовуватиме OpenAI GPT-3.5 для формування відповіді.

```
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    text = update.message.text
    user_id = update.message.from_user.id
    session_id = str(user_id)  # Використовуємо ідентифікатор користувача як SESSION_ID
    response = get_dialogflow_response(text, session_id)

    if response and not is_fallback_response(response):
        await update.message.reply_text(response)
    else:
        gpt_response = await get_gpt_response(text)
        await update.message.reply_text(gpt_response)
```

Рисунок 4.33. Функція «handle_message»

Далі створюємо функцію «get_dialogflow_response», яка надсилає запит до Dialogflow для отримання відповіді на основі введеного тексту.

```
def get_dialogflow_response(text: str, session_id: str) -> str:
    session_client = dialogflow.SessionsClient()
    session = session_client.session_path(DIALOGFLOW_PROJECT_ID, session_id)

    text_input = dialogflow.TextInput(text=text, language_code=DIALOGFLOW_LANGUAGE_CODE)
    query_input = dialogflow.QueryInput(text=text_input)

    try:
        response = session_client.detect_intent(session=session, query_input=query_input)
        return response.query_result.fulfillment_text
    except InvalidArgument:
        return None
```

Рисунок 4.34. Функція «get_dialogflow_response»

Наступним кроком буде створення функції «is_fallback_response», що відповідає за перевірку чи є відповідь від Dialogflow типовою фразою, яка вказує на відсутність розуміння запиту.

```python
def is_fallback_response(response: str) -> bool:
    # Перевіряємо, чи є відповідь від Dialogflow типовим повідомленням про відсутність відповіді
    fallback_phrases = [
        "Say that again?",
        "Sorry, could you say that again?",
        "Sorry, can you say that again?",
        "What was that?",
        "Can you say that again?",
        "Sorry, I didn't get that.",
        "I didn't get that. Can you say it again?",
        "I missed what you said. Say it again?",
        "Sorry, what was that?",
        "I didn't get that.",
        "I missed that.",
        "One more time?"
    ]
    return response in fallback_phrases
```

Рисунок 4.35. Функція «is_fallback_response»

Створюємо функцію «get_gpt_response». Вона буде надсилати запит до OpenAI GPT-3.5 для генерації відповіді.

```python
async def get_gpt_response(text: str) -> str:
    response = openai.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": text}
        ],
        max_tokens=150
    )
    return response.choices[0].message.content.strip()
```

Рисунок 4.36. Функція «get_gpt_response»

Далі реалізовуємо запуск чат-боту. Для запуску чат-боту використовується основна функція «main», яка створює додаток Telegram і запускає його в режимі полінгу. Ця функція забезпечує постійний моніторинг нових повідомлень і виклик відповідних обробників для взаємодії з користувачами.

```python
def main() -> None:
    application = ApplicationBuilder().token("6623136672:AAGCt2gCmssukglqaEstcq53d6iLGiv9Qgs").build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))

    application.run_polling()

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

Рисунок 4.37. Функція «main»

Перевіримо роботу чат-боту.

Для початку коли вводимо команду */start*, бачимо привітання.

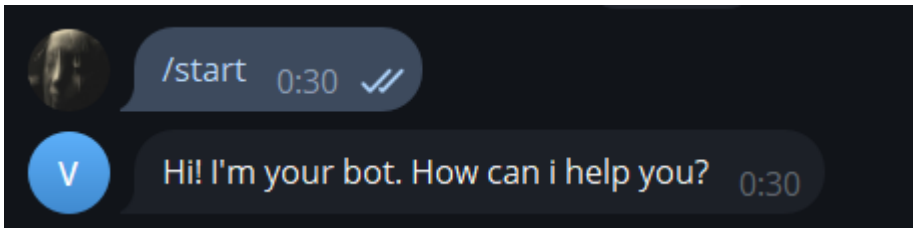

Рисунок 4.38. Вітання чат-боту

Далі спробуємо запитати у бота, щось, що є в сценаріях агенту «Small Talk», наприклад, «When were you born?» і побачимо, що він відповість фразою, яка була попередньо задана.

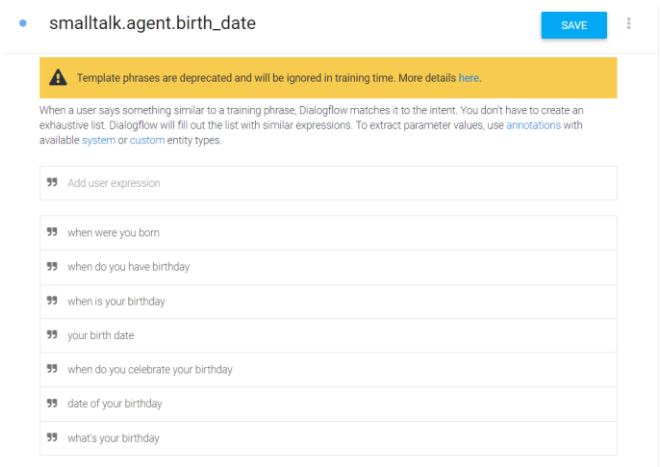

Рисунок 4.39. Задані фрази агента «Small Talk»

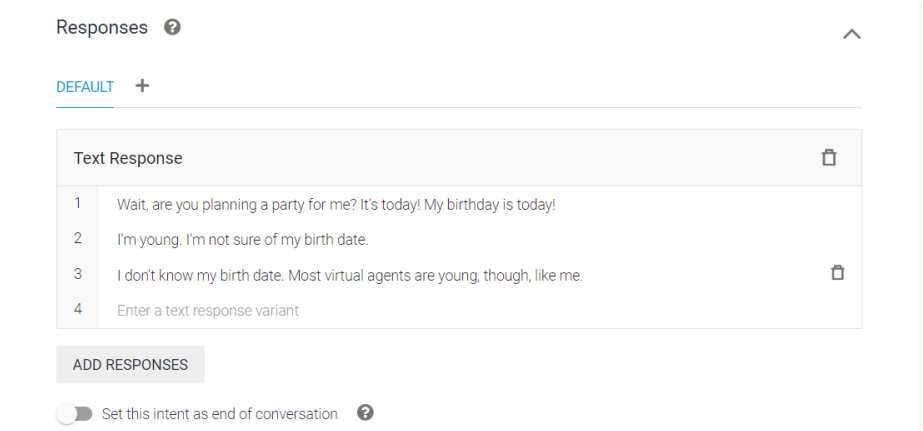

Рисунок 4.39. Задані відповіді агента «Small Talk»

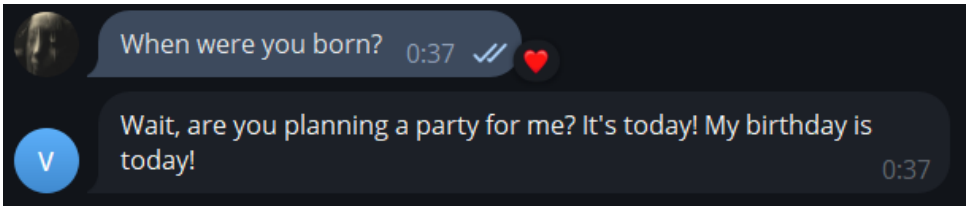

Рисунок 4.40. Відповідь чат-боту з використанням Dialogflow

Так як відповідь від Dialogflow отримується коректно, потрібно перевірити чи буде надходити відповідь від GPT-3.5. Для цього задаємо боту питання, якого немає прописаного в сценаріях агента «Small Talk» і побачимо, що він відповідає уже з використанням GPT-3.5.

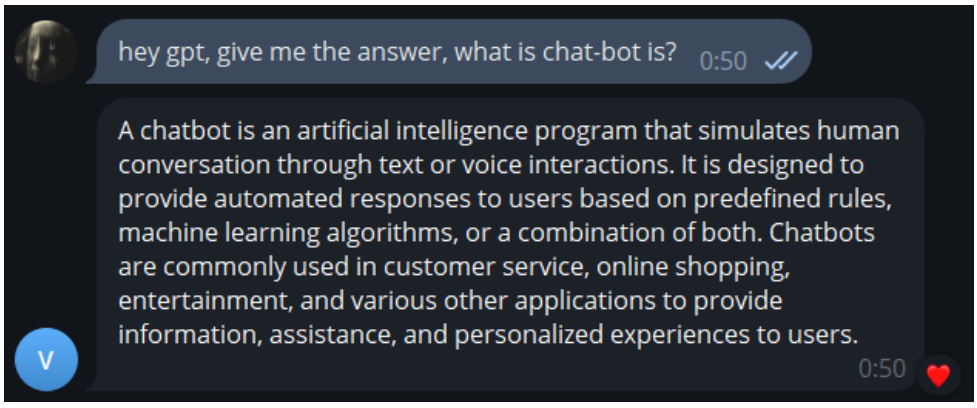

Рисунок 4.41. Відповідь чат-боту, з використанням GPT-3.5

Ось кілька прикладів функціональності чат-боту, з використанням GPT-3.5:

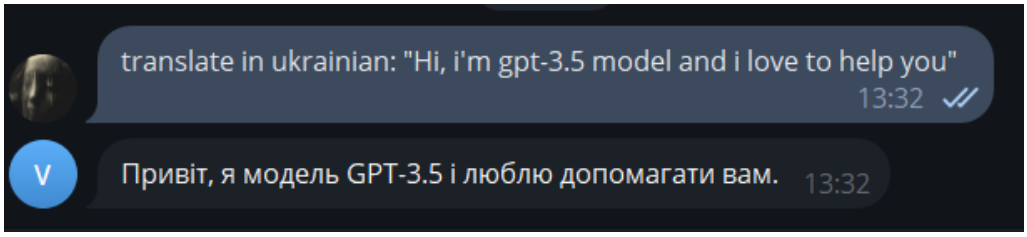

Рисунок 4.42. Переклад фрази чат-ботом, з використанням GPT-3.5

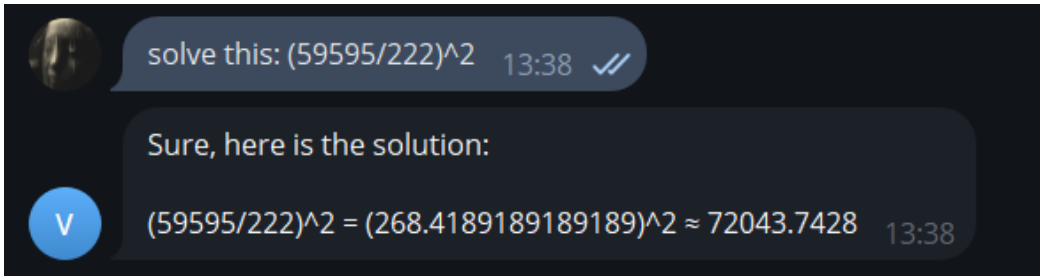

Рисунок 4.43. Розв’язання виразу чат-ботом, з використанням GPT-3.5

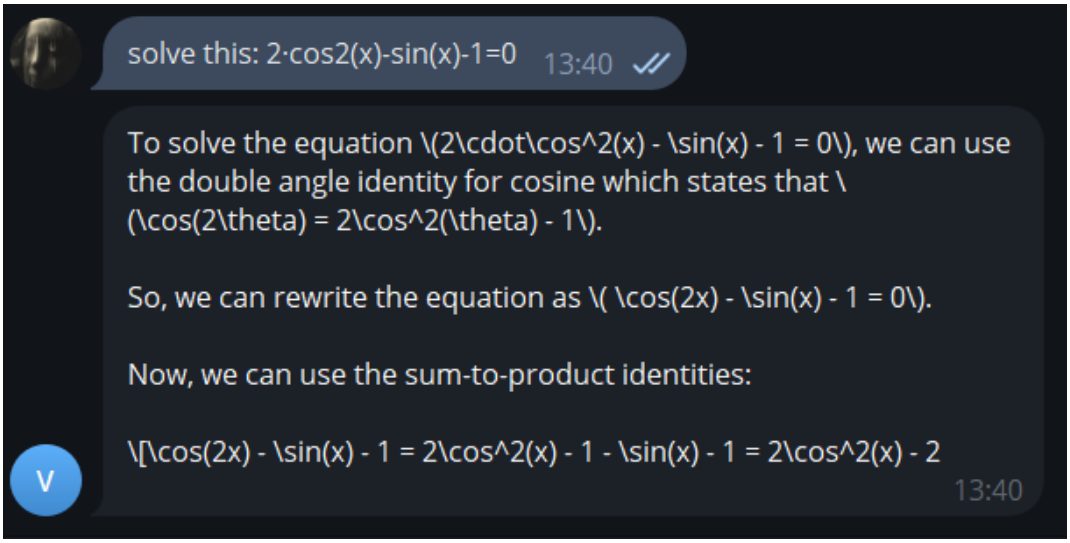

Рисунок 4.44. Розв’язання тригонометричного рівняння чат-ботом, з використанням GPT-3.5

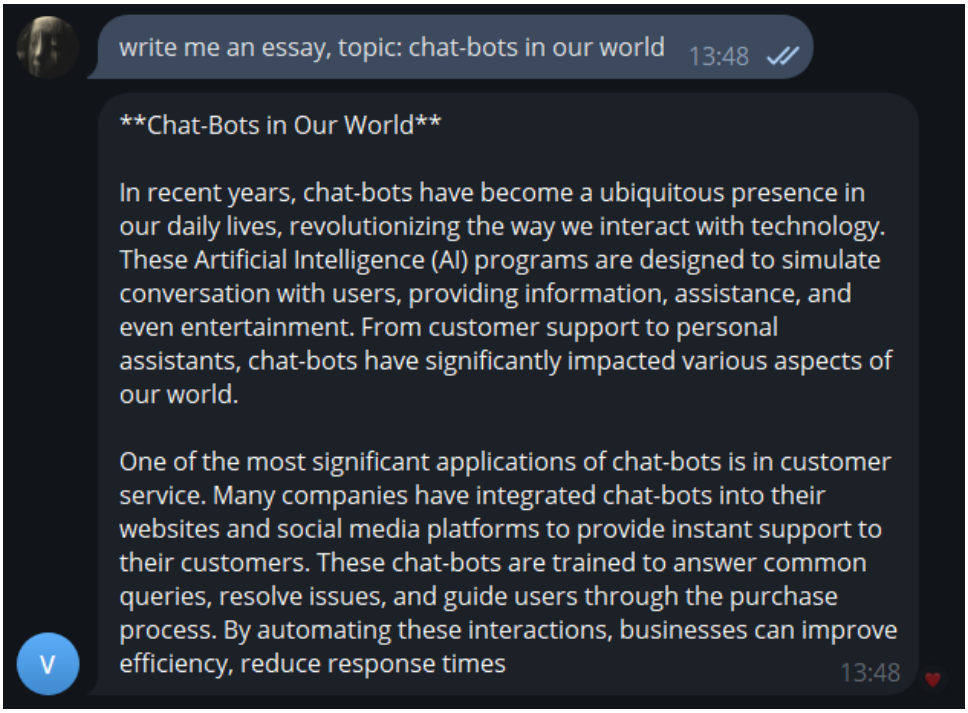

Рисунок 4.45. Написання ессе чат-ботом, з використанням GPT-3.5

Повний програмний код чат-боту можна переглянути у **Додатку А**.

ВИСНОВКИ

У ході проведеної дипломної роботи було здійснено розробку чат-бота для автоматизованого навчання користувачів з використанням контекстних моделей. Основною метою даного дослідження була розробка ефективного інструменту, який зможе забезпечити користувачам можливість отримувати швидкі та релевантні відповіді на їхні запити, використовуючи передові технології обробки природної мови.

У рамках роботи було реалізовано чат-бота, який використовує платформу Dialogflow для первинної обробки запитів користувачів. Dialogflow дозволяє розпізнавати інтенції та витягувати необхідну інформацію з текстових запитів, що забезпечує базовий рівень взаємодії з користувачами. Проте, у випадках, коли Dialogflow не може надати відповідь на запит користувача, в роботу вступає модель GPT-3.5, яка здатна генерувати відповіді на основі контексту запиту. Це дозволяє значно розширити функціональні можливості чат-бота та підвищити його ефективність у взаємодії з користувачами.

Реалізація чат-бота була здійснена на основі Telegram Bot API, що забезпечує зручність у використанні та доступність для широкого кола користувачів. Важливою особливістю розробленого чат-бота є його асинхронна архітектура, яка дозволяє ефективно обробляти запити користувачів у режимі реального часу, мінімізуючи затримки та забезпечуючи високий рівень продуктивності.

Під час розробки було проведено тестування та оптимізацію системи, що дозволило досягти високої точності відповідей та стабільної роботи чат-бота навіть при значному навантаженні. Використання сучасних технологій та інноваційних підходів у розробці забезпечило створення потужного інструменту для автоматизованого навчання користувачів.

Таким чином, у результаті виконання дипломної роботи було досягнуто поставленої мети – створено ефективний та надійний чат-бот для автоматизованого

навчання користувачів з використанням контекстних моделей. Реалізація даного проекту відкриває нові можливості для подальшого розвитку та впровадження подібних рішень у різних галузях, що потребують автоматизованої взаємодії з користувачами.

ДОДАТКИ

ДОДАТОК А.

Програмний код чат-бота

```
import os
import openai
from google.cloud import dialogflow_v2beta1 as dialogflow
from google.api_core.exceptions import InvalidArgument
from telegram import Update
from telegram.ext import Application, ApplicationBuilder, CommandHandler,
MessageHandler, filters, ContextTypes

# Налаштування OpenAI GPT
openai.api_key = 'sk-proj-qT49hTwSgjNCeikkIc4LT3BlbkFJg5iEN1cObrTCeh7Mzf52'

# Налаштування Dialogflow
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = 'C:/Users/Admin/Desktop/диплом/small-
talk1-9ftf-b87be9eb42b2.json'
DIALOGFLOW_PROJECT_ID = 'small-talk1-9ftf'
DIALOGFLOW_LANGUAGE_CODE = 'en-GB'

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text("Hi! I'm your bot. How can i help you?")

async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    text = update.message.text
    user_id = update.message.from_user.id
    session_id = str(user_id)  # Використовуємо ідентифікатор користувача як
SESSION_ID
    response = get_dialogflow_response(text, session_id)

    if response and not is_fallback_response(response):
        await update.message.reply_text(response)
    else:
        gpt_response = await get_gpt_response(text)
        await update.message.reply_text(gpt_response)

def get_dialogflow_response(text: str, session_id: str) -> str:
    session_client = dialogflow.SessionsClient()
    session = session_client.session_path(DIALOGFLOW_PROJECT_ID, session_id)

    text_input = dialogflow.TextInput(text=text,
language_code=DIALOGFLOW_LANGUAGE_CODE)
```

```
    query_input = dialogflow.QueryInput(text=text_input)

    try:
        response = session_client.detect_intent(session=session,
query_input=query_input)
        return response.query_result.fulfillment_text
    except InvalidArgument:
        return None

def is_fallback_response(response: str) -> bool:
    # Перевіряємо, чи є відповідь від Dialogflow типовим повідомленням про відсутність
відповіді
    fallback_phrases = [
        "Say that again?",
        "Sorry, could you say that again?",
        "Sorry, can you say that again?",
        "What was that?",
        "Can you say that again?",
        "Sorry, I didn't get that.",
        "I didn't get that. Can you say it again?",
        "I missed what you said. Say it again?",
        "Sorry, what was that?",
        "I didn't get that.",
        "I missed that.",
        "One more time?"
    ]
    return response in fallback_phrases

async def get_gpt_response(text: str) -> str:
    response = openai.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": text}
        ],
        max_tokens=150
    )
    return response.choices[0].message.content.strip()

def main() -> None:
    application =
ApplicationBuilder().token("6623136672:AAGCt2gCmssukglqaEstcq53d6iLGiv9Qgs").build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))
```

```
    application.run_polling()

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A brief history of Chatbots: https://chatbotslife.com/a-brief-history-of-chatbots-d5a8689cf52f [Електронний ресурс]
2. https://core.telegram.org/ [Електронний ресурс]
3. Bird S. Natural language processing with Python / S. Bird, E. Klein, E. Loper. – Sebastopol : O'Reilly. – 2009. – 504 p.
4. Jurafsky D. Speech and language processing / D. Jurafsky, J. H. Martin. – New Jersey : Prentice Hall. – 2023. – 628 p.
5. Russell S., Norvig P. Artificial Intelligence: A Modern Approach / S. Russell, P. Norvig. – New York : Pearson. – 2009. – 1159 p.
6. Kevin P. Murphy. Machine Learning: A Probabilistic Perspective / Kevin P. Murphy – New York : The MIT Press. – 2012. – 1104 p.
7. Coursera: Deep Learning Specialization: https://www.coursera.org/specializations/deep-learning [Електронний ресурс]
8. ChatGPT: Build a Chatbot with the new OpenAI API in Python!: https://www.udemy.com/course/chatgpt-build-a-chatbot-with-the-new-openai-api-in-python/ [Електронний ресурс]
9. YouTube: Chatbot Tutorials: https://www.youtube.com/watch?v=6GLFcm7dGiY&list=PLG9FQRMgm_JIFAQ1d6PxzARiySBupelaz [Електронний ресурс]
10. Бьорк, Йохан. Штучний інтелект: революція, яка змінює світ / Й. Бьорк, Л. Лавренс. – Київ: Видавництво «Основи», 2018. – 345 с.
11. Бутенкевич, В.М. Основи машинного навчання / В.М. Бутенкевич, І.А. Коваленко. – Київ: КНТ, 2019. – 270 с.
12. Коваль, О.О. Розробка чат-бота для навчання: огляд сучасних методів та підходів // Вісник Київського національного університету. – 2020. – №3. – С. 45-52.

13. Лебедєва, Т.П. Використання контекстних моделей у навчальних чат-ботах // Наукові записки Харківського університету. – 2021. – №6. – С. 12-19.

14. Пирогова, О. Чат-боти для навчання: як вони працюють і де їх використовують [Електронний ресурс] / О. Пирогова // Освітній портал. – Режим доступу: https://osvita.ua/innovations/chatbots-education. – Назва з екрану.

15. OpenAI. GPT-4 Technical Report [Електронний ресурс]. – Режим доступу: https://www.openai.com/research/gpt-4. – Назва з екрану.

16. Іваненко, С.В. Автоматизоване навчання з використанням чат-ботів: результати експериментальних досліджень // Матеріали Міжнародної науково-практичної конференції «Інноваційні технології в освіті». – Київ: НТУУ «КПІ», 2022. – С. 134-139.