Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики

Дослідження методів розробки чат-ботів

Кваліфікаційна робота

за спеціальністю «Інженерія програмного забезпечення» 121

Керівник кваліфікаційної роботи

Ст.в. Салата К.В.

_____*(підпис)*

«____» _____ 2023 р.

Виконав студент Дяченко М.Є.

«____» _____ 2023 р.

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Завідуючий кафедри мультимедійних систем,

доцент, кандидат наук

Жежерун О. П.

_____*(підпис)*

«____» _____ 2023р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студента 4 курсу факультету інформатики

Дяченка Максима Євгенійовича

Тема: Методи розробки чат-ботів

Зміст кваліфікаційної роботи:

- Вступ
- Розділ 1. Огляд предметної області чат-ботів
- Розділ 2. Платформа для розробки чат-ботів Dialogflow
- Розділ 3. Розробка чат-боту з архітектурою на основі інтентів з Dialogflow
- Список літератури

Дата видачі «____» _____ 2023 р.

Керівник_____*(підпис)*

Завдання отримав _____*(підпис)*

Календарний план виконання кваліфікаційної роботи:

Назва етапу курсової роботи	Термін виконання етапу	Примітка
Отримання завдання на кваліфікаційну роботу	10.11.2022	
Пошук джерел за темою роботи	05.02.2023	
Ознайомлення з джерелами за темою роботи	17.03.2023	
Написання теоретичної частини роботи	05.05.2023	
Створення функціонуючого чат-боту	15.05.2023	
Завершення документації процесу розробки	17.05.23	

Зміст

Вступ

Мета цієї роботи полягає в вивченні методів розробки чат-ботів та у практичному дослідженні розробки чат-ботів за допомогою Dialogflow. Актуальність теми полягає в стрімкому рості популярності месенджерів в спілкування між людьми та взаємодії з бізнесом, що створює необхідність в автоматизації частини цієї комунікації.

Чат-боти є програмами, що імітують спілкування з живою людиною. Такі програми можуть виконувати багато функцій, наприклад, забезпечення зв'язку з клієнтами, інформування про певні події, перегляд і купівлю товарів чи послуг. Вони забезпечують цілодобову лінію зв’язку і автоматизують велику частину комунікаційних процесів, що допомагає бізнесу збільшувати ефективність та прибуток. Станом на сьогодні, використання чат-ботів є базовою необхідністю у будь-якій сфері, яка потребує активної комунікації з клієнтами.

В цій роботі буде наведено аналіз предметної області чат-ботів та детальний опис програмного забезпечення Dialogflow, як одного з популярних інструментів розробки, вузько спеціалізованого саме під створення чат ботів. Буде описаний принцип архітектури чат-ботів на основі розпізнання намірів користувача, притаманний платформі Dialogflow. Буде створено Telegram-бота з функціоналом збереження інформації про події з використанням Dialogflow.

Розділ 1. Огляд предметної області чат-ботів

1.1 Визначення поняття чат-бот

Як вже було сказано у вступі, чат-боти – це програмами, що дозволяють імітувати розмову з людьми. Термін «чат-бот» був вперше використаний у 1994 році. За рахунок здатності імітувати діалоги, чат-боти застосовуються у різних галузях, таких як електронна комерція, розваги, освіта та інші.

Зазвичай, чат-бот пропонує користувачу обрати варіант серед запропонованих. Тобто отримання інформації або виконання інших задач відбувається шляхом вибору запропонованих відповідей, або запропонованих категорій, і користувач не вводить текст до вікна чату. Більш складні боти надають можливість вводу тексту і отримування інформації відповідно до запиту користувача. [1]

1.2 Роль технологій штучного інтелекту в розробці чат-ботів

Технології штучного інтелекту відіграють важливу роль у розробці чат-ботів. Основні технології, які використовуються в цій області, включають машинне навчання та обробку природньої мови. Прогрес у машинному навчанні значно покращив продуктивність використання штучного інтелекту у багатьох сферах. З розвитком, штучний інтелект почав перевершувати людські можливості в виконанні вузькоспеціалізованих задач. Найвідомішим прикладом, що сколихнув світове сприйняття прогресу ШІ, є комп'ютер для гри в шахи «Deep Blue», створений компанією IBM. У 1997 році він здобув перемогу над чемпіоном світу Гаррі Каспаровим. Використовуючи технології навчання з підсиленням (англ. reinforced learning - навчання програми вибирати найкращі дії в певному середовищі для отримання максимальної нагороди), компанія OpenAI розробила штучний інтелект для гри в Dota2, якому вдалося перемогти тогочасних чемпіонів світу – команду OG. [2]

Можливості ШІ показувати результати на рівні людських супроводжується все частішим використанням слова «розуміння» в цьому контексті. Розуміння природної мови, РПМ (англ. natural-language understanding, NLU) - підтема обробки природної мови, що займається машинним розумінням прочитаного. Застосування цих технологій дозволяє чат-ботам розпізнавати та розуміти повідомлення користувачів, а також надавати їм відповіді в реальному часі. Говорячи про розуміння, вживаються такі поняття, як «широта» та «глибина». «Широта» системи вимірюється розміром її словникового запасу та граматики. «Глибина» вимірюється ступенем, до якого її розуміння наближається до розуміння носія мови.

З технічної точки зору, розробка систем РПМ потребує знання лексикону мови, та парсер з набором граматичних правил, що позволить приводити речення природньої мови в їх формалізовану репрезентацію. Ключовими є поняття синтаксису та семантики.

Синтаксис описує правила, за якими слова та фрази можуть бути складені в речення. Він визначає, як граматично вірно сформулювати думку, використовуючи мову. Семантика, з іншого боку, описує значення слів та фраз, тобто їх смисл. Вона визначає, яким чином слова та фрази співвідносяться між собою та які концепти вони виражають. Наприклад, слово "дерево" має семантичний зв'язок з такими концептами, як "рослина", " корінь ", "листя" тощо.

Використовуючи машинне навчання та спираючись на множину семантичних зв'язків та синтаксичних правил, вдається досягти високий відсоток розуміння природної мови, що позволяє чат-ботам розуміти не лише чітко задані команди, а й вловлювати однаковий сенс в багатьох синонімічних повідомленях від кінцевого користувача.

1.3 Класифікація чат-ботів за принципом роботи

За особливостями структури, чат-ботів можна класифікувати за наступними характеристиками [4]:

1. Режим взаємодії (на основі тексту або на основі голосу/мовлення).
2. Розуміння команд або розуміння природної мови.
3. Орієнтований або не орієнтований на завдання.

Режим взаємодії з чат-ботом, як правило, визначається на основі практичної сфери його застосування. Голосовий режим взаємодії потребує додаткових зусиль при розробці, оскільки до алгоритму роботи додається необхідність розпізнання голосу з подальшою обробкою та перетворенням в відповідний текст. До того ж, існує багато випадків, в яких текстове управління буде зручнішим за голосове. Як наслідок, більшість чат-ботів працюють на основі текстової взаємодії. Для кращого розуміння специфіки вибору режиму взаємодії, розглянемо один з випадків використання голосового вводу. Amazon Echo — стаціонарна колонка, з встановленим голосовим помічником Amazon Alexa. Функції пристрою включають голосову взаємодію, відтворення музики, створення списків спав та будильників та керування кількома розумними пристроями, діючи як центр домашньої автоматизації. Найголовніший фактор, що надає змогу використовувати голосове керування для цього пристрою – приватне середовище, в якому людина зможе вільно давати команди та нівелювати вплив стороннього шуму, який може завадити голосовому асистенту зрозуміти команду, а користувачу - почути відповідь. До того ж, актуальність саме голосовому вводу надає те, що вдома людина не обов’язково фізично знаходиться біля пристрою, з якого можна вводити команди чи запитання. На противагу цьому можна надати спілкування зі службою підтримки – завдання, яке також може бути частково автоматизоване за рахунок чат-ботів. В цьому випадку, людина є налаштованою до розмови, та гарантовано має доступ до приладу, з якого буде вести комунікацію. У підсумку, можна сказати, що для більшості випадків можна надати перевагу саме

текстовій взаємодії з чат-ботом, однак голосова взаємодія є набагато зручнішою для обмеженого набору випадків, що сильно залежать від навколишнього середовища.

Орієнтовані на завдання в чат-боті можна характеризувати як такі чат-боти, що спеціалізуються лише на виконанні певної задачі, наприклад, бронювання білетів, пошук товарів тощо. Чат-боти, орієнтовані на завдання, добре працюють у обмежених доменах. Ці чат-боти не володіють загальними знаннями, і ви не можете запитати в них що завгодно чи вести діалог, подібний до живого спілкування. Використання таких чат ботів є ефективним для досягнення конкретної мети. [5]. Відповіді на команди (як і самі команди) можуть бути чітко прописаними розробниками, що позволяє створити такого чат-бота без використання алгоритмів розуміння мови та машинного навчання. Чат-боти, не орієнтовані на завдання, створені для максимального уподібнення до повноцінного діалогу. Для досягнення цієї цілі, використовуються технології машинного навчання та штучного інтелекту. Найвідомішим прикладом такого чат-боту є ChatGPT, що використовує статичну модель мови, та алгоритми навчання з підсиленням.

1.4 Класифікація чат-ботів за сферами використання

Розібравшись з технічною класифікацією чат-ботів, варто також покрити до практичних випадків їх використання в різних галузях. Станом на сьогодні, існує велика різноманітність чат-ботів за їхнім функціоналом. Як було вказано у змісті, чат-боти часто використовуються в бізнесі - b2c (business to client) взаємодія, але це є далеко не єдиною областю використання чат-ботів. До того ж, в b2c-взаємодії також існують розгалуження у використанні.

- Використання в бізнесі: комерційний інтерес до технології чат-ботів є одним з основних рушіїв їх розвитку. Вони можуть бути використані для різноманітних бізнес-завдань, таких як надання інформації про продукти або послуги, обробка замовлень, надання відповідей на питання клієнтів, рекламна розсилка. Такі чат-

боти можуть бути реалізовані в месенджерах та бути інтегрованими в веб-сайт у вигляді спливаючого вікна з чатом. або за допомогою Інформація з переписок з чат-ботом може збиратись, аналізуватись та використовуватись для загального покращення якості послуг. Також, боти можуть бути інтегровані в системи для управління відносин з клієнтами (з англ. CRM – Customer relationship management) для автоматизації частини процесів.

- Створення утиліт для месенджерів: за рахунок чат-ботів, можна створювати утиліти в месенджерах для виконання додаткових функцій, не передбачених платформою. Такі боти можуть виконувати різноманітні функції, такі як перегляд погоди, створення нотаток з нагадуваннями та багато іншого. Можливість звернення до чат-ботів в месенджері є зручним та швидким спосіб здійснення різноманітних операцій для кінцевого користувача. До того ж, самим месенджерам вигідна наявність активної спільноти розробників, яка покращує досвід користування інших своїм продуктом. Саме за рахунок чат-боту, в месенджері Telegram можна було створювати опитування до того, як цей функціонал був доданий в офіційному оновленні.
- Розважальні чат-боти: частина з існуючих чат-ботів була створення задля виконання розважальних функцій. За допомогою текстового інтерфейсу чат-ботів можливо написати примітивні покрокові ігри, такі як хрестики-нулики.
- Обробка контенту, пропонованого користувачами: такий підхід збору інформації від великої публіки позволяє налаштувати захист від спаму, налаштувати обмеження при вводі даних, та приховати за фасадом боту контактні дані людини, що збирає інформацію. Велика кількість телеграм-каналів використовує таких ботів для публікації запропонованого користувачами контенту.

1.5 Переваги застосування чат-ботів

Використання чат-ботів набуло великої популярності протягом останніх років, при тому що вони, як правило, не почали виконувати цілком нові функції. Натомість, з їх допомогою вдалося частково автоматизувати чи зробити значно зручнішими велику кількість бізнес-процесів. Для розуміння цього переходу, та чому він став успішним, варто порівняти використання чат-ботів зі старішими підходами до вирішення задач в різних сферах діяльності.

Досить часто боти використовується для надання інформації про продукт або сервіс, що допомагає зменшити навантаження на співробітників в службі підтримки. В цьому контексті, значною перевагою є надаваний чат-ботами швидкий і безперервний доступ до інформації і послуг, незалежно від часу доби та місцезнаходження користувачів, а також його здатність обробляти багато запитів одночасно.

Більшість питань клієнта є досить шаблонними. Наприклад, найчастішими питаннями до сервісу доставки будуть питання по типу «як дізнатися стан посилки?», «скільки необхідно чекати отримання». Чат-бот зможе легко надавати відповіді на подібні питання, а інформацію в відповідях буде легко оновити. До того ж, чат-боти можуть бути легко інтегровані з іншими сервісами, за рахунок чого значно розширюється їх функціональність. Повертаючись до прикладу з сервісом доставки, може бути надана можливість для користувача ввести номер замовлення, по якому він отримає відповідь зі статусом його замовлення. Для зручності, найчастіші питання можна структурувати в меню чат-бота. Користувач зможе використовувати це меню для швидкої та інтуїтивно зрозумілої навігації по різних питаннях для отримання бажаної відповіді.

З наведених вище аргументів випливає те, що вигідною інвестицію часу та ресурсів в розробку, оскільки зменшує кількість необхідного персоналу служби підтримки, хоча взаємодія з людиною все ще залишається незамінною для деяких

більш складних випадків. На додачу, з’являється значна перевага у вигляді надання цілодобового сервісу, що позволить користувачам отримати відповіді на свої питання в неробочі години.

Висновок до розділу 1

В цьому розділі була детально описана предметна область чат-ботів. Надано опис технології розуміння штучної мови (РПМ), та описано їх ключову роль для розвитку технології чат-ботів. Надано класифікацію чат-ботів за функціоналом та принципом роботи, обґрунтовано доцільність та переваги їх використання.

Розділ 2. Платформа для розробки чат-ботів Dialogflow

Dialogflow — це платформа для розуміння природної мови (NLU), яка використовується для розробки та інтеграції розмовного інтерфейсу користувача в мобільні додатки, веб-додатки, пристрої та чат-боти. Окрім роботи з текстовими повідомленнями, позволяє розпізнавати голос. Платформа, випущена в 2014 році під назвою Api.ai, була придбана компанією Google у 2016 році.[6] У 2017 році перейменована на Dialogflow та стала частиною Google Cloud Platform [7].

В вересні 2020 року компанія Google доповнила сервіс, створивши нову платформу Dialogflow CX (Customer Experience). Попередня версія була перейменована в Dialogflow ES (Essentials). Платформи використовують різні парадигми роботи з чат-ботами. Dialogflow ES базується на зберіганні контексту та розпізнаванні інтенту (анг. Intent – намір) користувача, в той час як Dialogflow CX використовує підхід, подібний до скінченних автоматів, та запам’ятовує стан користувача. Схеми діалогу в Dialogflow CX будується з візуальних блоків, що контролюють стани розмов (Рис 2.1). Даний підхід спрощує розробку великих проектів, що потребують імплементацію великих та розгалужених схем діалогу. Dialogflow ES має безкоштовні тарифні плани, в той час як Dialogflow CX є виключно платним сервісом. В рамках даної роботи, використовуватиметься Dialogflow ES, оскільки практична частина не вимагає створення великих та розгалужених сценаріїв розмов. Також, подальші використання назви Dialogflow позначатимуть саме Dialogflow ES.

Технології NLU є вбудованими в платформу та допомагають розпізнавати користувацький запит. Розробник може перевіряти правильність розпізнавання запитів, та завантажити текстовий документ з набором запитів, що використовуватиметься для навчання чат-боту. Також, існує можливість розпізнавання слів, в яких допущено граматичну помилку.

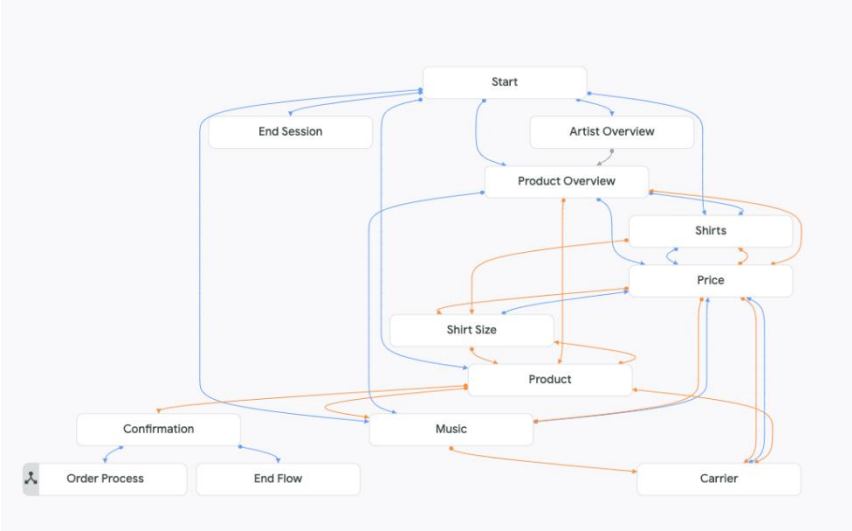

Рисунок 2.1. Візуальна репрезентація діалогу в Dialogflow CX [8]

Платформа Dialogflow позволяє розробляти чат-ботів з використанням безсерверної архітектури, побудова сценаріїв розмови будується через користувацький інтерфейс. Через користувацький інтерфейс є можливість налаштувати усі діалоги, що проходять у форматі «питання-відповідь», та не потребують обчислень чи взаємодій зі сторонніми сервісами. Для виконання коду наявна можливість використання Google Cloud Functions. Використання хмарних функцій є платним. Також, існує можливість обробляти повідомлення користувача на власному сервері за допомогою вебхуків.

Існують інтеграції з багатьма найпопулярнішими месенджерами та сервісами телефонії, серед яких є Facebook Messenger, Slack, Telegram та інші, та можливість інтегрувати сторінку чи вікно чату в власний веб-сайт. Також, платформа підтримує велику кількість мов, що позволяє використовувати Dialogflow для розробки чат-ботів на широку аудиторію.

Перелічивши основні можливості платформи, можна виділити її переваги:

- Зручний користувацький інтерфейс для створення сценаріїв діалогу.

- Використання хмарних обчислень на серверах Google, можливість використання serverless архітектури.
- Використання машинного навчання на основі користувацьких запитів.
- Вбудовані алгоритми розуміння людської мови.
- Інтеграції з найпопулярнішими месенджерами, багатомовність.

2.1 Основні поняття та концепти при роботі з Dialogflow

В цій частині роботи будуть детально описані основні методи, поняття та налаштування, що використовуються при розробці чат-ботів в даній платформі – а саме: Agents, Intents, Entities, Fulfillment, Training, Validation.

2.1.1 Агенти (Agents)

Агент в Dialogflow – це віртуальний агент, який обробляє одночасні розмови з кінцевими користувачами. Для пояснення цього терміну, можна привести паралель з людиною – робітником кол-центру.[9] Агент має бути підготованим для надання відповідей на деякий обмежений набір запитань. По суті, кожен агент є окремим проектом чат-боту. Dialogflow позволяє створювати декількох агентів для виконання різних задач. На рівні агенту здійснюються базові налаштування, такі як мови та назва агенту, вибір часового поясу (використовується для обробки дат), налаштування середовищ та доступу до проекту.

2.1.2 Інтенти (Intents)

Інтент (намір) - класифікує намір кінцевого користувача для одного ходу розмови. Для кожного агента визначається багато намірів, в свою чергу, набір усіх намірів може обробляти повну розмову. Коли кінцевий користувач надсилає тектове чи голосове повідомлення, Dialogflow зіставляє вираз кінцевого користувача з усіма

наявними інтентами, та визначає найбільш підходящий з них для надання відповіді. [10]

Для розпізнавання інтенту необхідно ввести набір слів чи речень, що можуть бути сказані користувачем для отримання бажаного результату. Цей набір називається тренувальні фрази. Як приклад, візьмемо випадок, коли користувачу необхідно дізнатися погоду. Для цього запиту, тренувальними фразами можуть бути «яка зараз погода?», «скільки зараз градусів?», тощо. Немає необхідності надавати список усіх можливих фраз, оскільки Dialogflow розпізнаватиме схожі фрази не лише за повним збігом. Однак, більший набір тренувальних фраз покращуватиме розуміння інтенту. Існує можливість повного відключення алгоритмів РПМ та машинного навчання для певних інтентів. Тоді, чат-бот буде відповідати тільки на повідомлення, що повністю збігаються з однією з тренувальних фраз. Крім тренувальних фраз, необхідно задати відповідь чат-боту на кожен інтент. Є можливість додавання багатьох відповідей, що матимуть різний вигляд, але однаковий зміст, що дозволяє збільшувати схожість комунікації з ботом на живе спілкування.

Рисунок 2.1.2.1 Приблизна структура інтенту привітання

Окрім інтентів, створених користувачем, під час створення агента автоматично створюються два інтенти:

- Інтент привітання за замовчуванням (default welcome intent): спрацьовуватиме, коли користувач надсилає перше повідомлення у чат-боті. Відповіддю на цей інтент може бути вітання та короткий опис основного функціоналу у чат-боті.

- Резервний інтент за замовчуванням (default fallback intent): спрацьовуватиме у випадках, коли не вдалось розпізнати інтент повідомлення користувача. Відповідь за замовчуванням складається з набору статичних текстових відповідей, по типу «Я не зрозумів ваше повідомлення, напишіть, будь ласка, ще раз». [11]

2.1.3 Сутності (Entities)

Сутність - тип параметру в запиті, що визначає як саме вилучаються дані з виразу кінцевого користувача.[12] За рахунок сутностей, можна класифікувати дані, що були введені користувачем, віднести їх до певного типу та зберегти значення, як змінну в інтенті. Dialogflow за замовчуванням підтримує сутності, які використовуються найчастіше, такі як дати, адреси, номери телефону та багато інших.

Продовжуючи приклад з погодою з попередньої частини, ми можемо доповнити інтент користувача, додавши уточнення стосовно міста, про погоду в якому необхідно дізнатися. В такому випадку, запит виглядатиме як «яка зараз погода в місті Київ?». Для обробки таких запитів, ми можемо виділити назву міста в полі для редагування тренувальних фраз, та присвоїти йому тип «@sys.location». Отримавши дані про тип, Dialogflow розпізнаватиме й інші міста, буде створена змінна, що зберігатиме цей параметр. В подальшому, його може бути використано в запиті на сторонній сервіс, для отримання погоди.

```
{
  intent_name: "weather",
  training_phrases: [ "Weather in $[Kyiv]"],
  entities: [{name: "city", type: "@sys.location"}],
  responses: [
    "Weather in $[city] is just fine ;)"
  ],
  context: ""
}
```

Рисунок 2.1.3.1 Приблизна структура інтенту з розпізнанням сутності

Крім системних сутностей, платформою підтримується можливість створення власних сутностей. При створенні сутності, розробник має вручну вводити її можливі значення та синоніми до цього самого значення. Як і у випадку з інтентами, Dialogflow може автоматично розширювати синонімічний ряд для нових сутностей.

2.1.4 Виконання коду (Fulfillment)

Для переважної більшості чат-ботів є необхідним виконання додаткової логіки, що потребує виконання додаткового коду. Даний функціонал підтримується платформою Dialogflow в розділі Fulfillment. Існує можливість підключення вебхуку (webhook) або використовувати вбудований редактор коду. При роботі з Fulfillment, Dialogflow відправлятиме запит з повідомленням користувача. Окрім тексту повідомлення, тіло запиту міститиме усю додаткову інформацію про повідомлення в форматі JSON (JavaScript Object Notation), таку як інтент, змінні, отримані за допомогою сутностей тощо. Після обробки запиту, необхідно повернути відповідь у JSON-форматі, що відповідатиме структурі відповідей Dialogflow. Використання Fulfillment регулюється на рівні кожного інтенту, тобто виконання коду можна вмикати лише в необхідних для цього місцях.

Вебхук – це метод збільшення або розширення функціональності веб застосунку за допомогою користувацьких зворотних викликів (callbacks), що можуть обслуговуватися або керуватися користувачами або веб розробниками, не обов'язково пов'язаними з вищезгаданим веб застосунком. [13] Окрім раніше згаданих вимог до обробки запитів, URL вебхуку повинен бути в публічному доступі та підтримувати HTTPS запити. Також, з міркувань безпеки, в Dialogflow реалізована можливість використання базової аутентифікації та додавання хедерів.

Вбудований редактор коду в Dialogflow використовує середовище виконання Nodejs 10, з можливістю завантажувати сторонні бібліотеки, вказані в package.json,

при розгортці чат-боту. Для виконання коду застосовуються хмарні обчислення на платформі Google Cloud Functions.

Використання хмарних функцій позволяє повністю слідувати парадигмі serverless застосунків, але накладає значні обмеження при розробці, оскільки Nodejs є єдиним наявним середовищем виконання. В цей час, вебхуки можуть бути імплементовані з використанням будь-якої мови програмування та розміщуватись на власних серверах.

2.1.5 Відладка помилок та тренування

Ключовим елементом розробки програмного забезпечення, і, зокрема, чат-ботів є тестування та відладка застосунку. В цій частині роботи, буде описана специфіка тестування та відладки помилок в Dialogflow, а саме - використання даною платформою алгоритмів природного розуміння мови для розпізнавання інтентів, від чого залежить те, чи буде надана правильна відповідь користувачу. Наслідком помилки при розпізнаванні інтенту буде надання некоректної відповіді.

Найчастіше, такі помилки виникають за рахунок обмеженої кількості тренувальних фраз. Dialogflow надає функціонал тренування процедури розуміння інтентів на основі історії переписок у вкладці Training. В ній, розробник може переглядати історію листування з чат-ботом, та побачити інтент, до якого належить кожне з повідомлень. У випадку, якщо інтент розпізнано коректно, текст повідомлення можна додати до списку тренувальних фраз. Якщо нерелевантне по відношенню до жодного з наявних інтентів повідомлення було помилково розпізнано, його можна додати до тренувальних фраз резервного інтенту за замовчуванням, як негативний приклад.

Крім валідації повідомлень з історії, існує можливість завантажити власний текстовий файл (у форматі «.txt») чи архів з файлами з користувацькими фразами з метою тренування. Після завантаження, повідомлення відображатимуться в історії, і

зможуть бути перевіриними на правильність. Для пошуку помилок в Dialogflow існує вкладка Validation, де будуть відображені всі помилки, що виникали за час роботи чат-боту.

Алгоритми машинного навчання та РПМ платформи Dialogflow не є у відкритому доступі, та для розробників доступний лише обмежений набір налаштувань. Можливо змінити поріг класифікації наміру. Під час пошуку відповідного наміру Dialogflow оцінює потенційні збіги за допомогою достовірності виявлення наміру, також відомої як оцінка достовірності. Ці значення коливаються від 0,0 (повністю невизначений) до 1,0 (повністю впевнений). [14] Якщо намір із найвищою оцінкою має оцінку достовірності, що перевищує або дорівнює порогу класифікації, вона повертається як збіг. Якщо жодні наміри не відповідають пороговому значенню, виконується резервний намір. [15]

Крім цього, наявна автоматична корекція граматичних помилок при розпізнаванні повідомлення, ввімкнення/вимкнення автоматичної валідації. Також, для оптимізації роботи з великими агентами, можливо вимкнути автоматичне тренування при кожній збереженій зміні.

2.2 Загальна схема архітектури на основі інтентів

Визначивши базові поняття, ми можемо перейти до визначення загальних принципів архітектури на основі інтентів. Звертаючись до наведеної в першому розділі класифікації, Dialogflow може працювати як з текстовим, так і з голосовим вводом. Чат-боти, створені за допомогою цієї платформи, є орієнтованими на певну задачу, та застосовують алгоритми NLP для покращення роботи.

Для розуміння архітектури на основі інтентів, розглянемо кроки, що відбуваються для отримання відповіді, починаючи з етапу отримання повідомлення:

1. Повідомлення від користувача отримано.
2. Dialogflow визначає інтент на основі тренувальних фраз кожного з інших інтентів, що найкраще підходить для цього повідомлення (рис. 2.2.1)
3. (Опційно) Зберігається новий вихідний контекст, у якому перебуватиме чат-бот, коли буде отримувати наступне повідомлення. Тільки інтенти, у яких визначено цей же вхідний контекст, зможуть обробляти це нове повідомлення.
4. З повідомлення виокремлюються сутності та зберігаються у спеціально створені для них змінні (рис. 2.2.2).
5. (Опційно) Запит обробляється за допомогою хмарної функції або вебхуку.
6. Надсилається результат повідомлення.

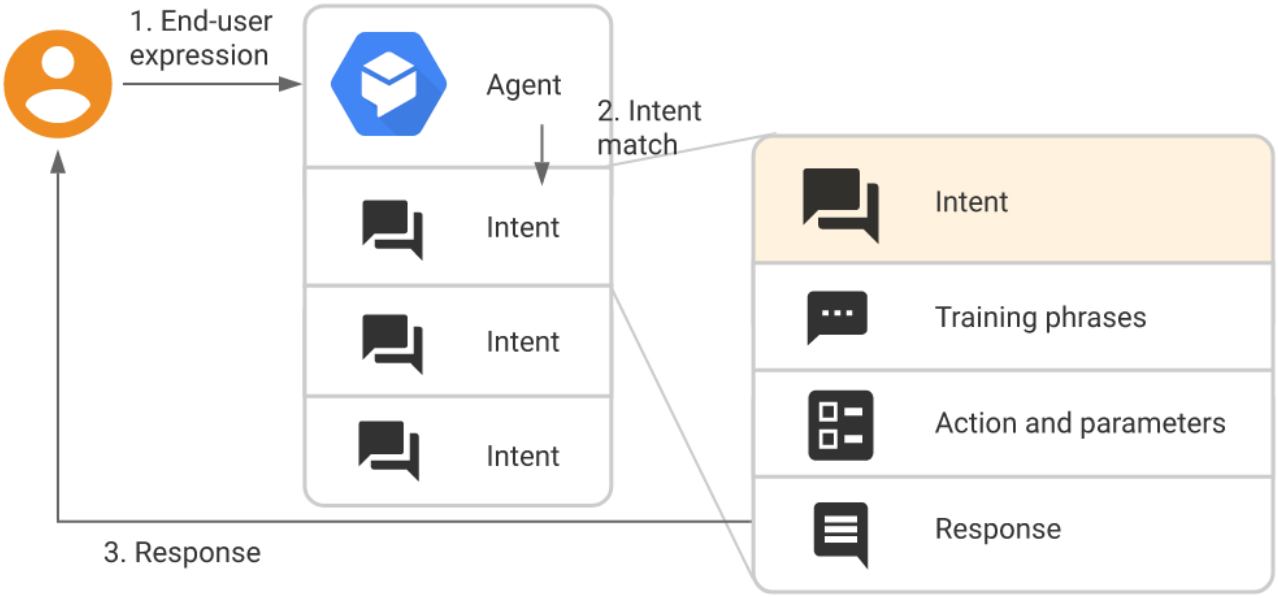

Рисунок 2.2.1 Розпізнавання інтентів [9]

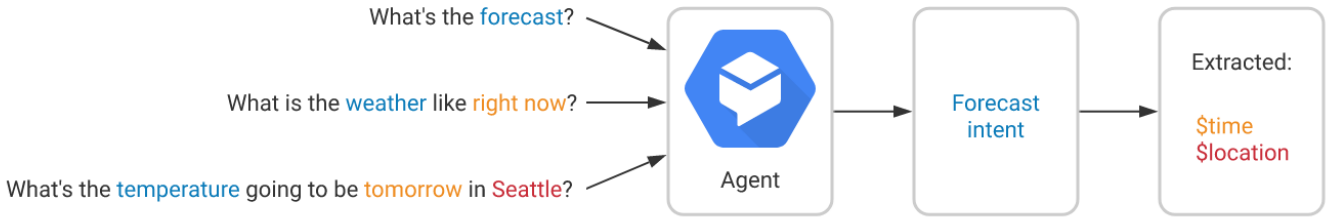

Рисунок 2.2.2 Розпізнавання сутностей [9]

Архітектура чат-ботів на основі інтентів пропонує кілька переваг. Вона спрощує взаємодію з ботом та покращує користувацький досвід за рахунок використання технологій РПМ, підтримує взаємодію як на основі правил, так і з довільними повідомленнями. Однак ця архітектура також має недоліки, такі як керування численними намірами та їх варіаціями, потенційна втрата точності для складних запитів і обмеження в контекстному розумінні користувача.

Висновок до розділу 2

В цьому розділі було розглянуто основний функціонал Dialogflow, що використовуватиметься у розробці чат-боту. Надано опис понять «інтент», «сутність», описаний процес виклику коду та процесу тренування чат-боту. Детально описано принцип архітектури чат-ботів на основі інтентів, перелічено її переваги та недоліки.

Розділ 3. Розробка чат-боту з архітектурою на основі інтентів з Dialogflow

В даному розділі буде описано розробку чат-бота в месенджері Telegram з використанням платформи Dialogflow, що виконуватиме функцію помічника для збереження інформації про розпорядок дня користувача. У якості бази даних, необхідної для збереження даних, введених користувачем в процесі переписки з чат-ботом, використовуватиметься Firebase. Для взаємодії з базою даних, буде створено вебхук з використанням середовища виконання Nodejs з мовою JavaScript.

3.1 Технічне завдання чат-бота

Необхідно реалізувати наступний функціонал:

- Зберігання інформації про події (назва, дата, час, регулярність).
- Додавання нової події.
- Видалення події.
- Перегляд списку подій в будь-яку дату або за найближчий тиждень.
- Отримування нагадувань про події.

Формат даних для додавання події:

- Назва події - текстове поле.
- Дата та час – текстове поле, конвертується в дату у форматі «dd.MM.yyyy HH:mm» за допомогою Dialogflow.
- Тривалість події – текстове поле, конвертується в об’єкт за допомогою Dialogflow.
- Регулярність – наскільки часто подія повторюється – одноразова подія, відбувається раз в тиждень чи раз в місяць.

Нефункціональні вимоги:

- Розпізнавання інтентів та сутностей, а також збереження структури можливих діалогів здійснюється за допомогою Dialogflow.
- Використання Firebase для збереження даних.
- Для взаємодії з базою даних задля збереження чи видалення інформації, використати вебхук, створений на Nodejs.
- Текстовий інтерфейс чат-боту доступний в менеджері Telegram.

Використання архітектури на основі інтентів підходить для цього завдання, оскільки воно не потребує великої розгалуженості сценаріїв діалогу. Також, парсинг таких даних як дати та час з допомогою РПМ є набагато зручнішими, ніж ручне введення.

3.2 Створення чат-боту в Telegram та інтеграція з Dialogflow

Важливим аспектом при створенні чат-боту є вибір платформи для його розміщення. Найчастіше, чат-ботів створюють в рамках вже існуючих месенджерів. Для цієї роботи було обрано месенджер Telegram. Dialogflow підтримує інтеграцію з даним застосунком, що позволить повною мірою розкрити можливості для інтеграції.

Для створення чат-бота в Telegram потрібно використати іншого бота з назвою "BotFather". Після введення команди "/newbot", ми можемо ввести ім'я та унікальний ідентифікатор (юзернейм) для URL-посилання на нього. Після цього ми отримуємо унікальний токен бота (приклад токену: «123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11»), який дає доступ до API. Крім того, через "BotFather" можна налаштувати бота, встановити зображення профілю, опис, змінити ім'я тощо.

Після отримання токену, необхідно налаштувати інтеграцію з платформою Dialogflow. Для цього, необхідно перейти в розділ Integrations, відкрити меню інтеграції з Telegram та скопіювати токен, отриманий з BotFather.

3.3 Надання довідкової інформації про чат-бота

В незалежності від специфіки чат-боту, типовим є наявність можливості перегляду його основних функцій, інструкції по використанню тощо. Хорошою практикою є надання короткого опису та інструкції по використанню на початку діалогу. Для цього, необхідно створити окремий інтент для обробки старту листування. Особливістю чат-ботів в Telegram є те, що перше повідомлення, введене користувачем завжди є командою «/start». Для обробки цього повідомлення, можна використати одну з подій в Dialogflow – TELEGRAM_WELCOME. Подія в Dialogflow виступає аналогом тренувальних фраз при розпізнанні інтенту. Інтенти викликаються, коли відбуваються специфічні для платформи події. Аналогічним способом, можна створити інтент для обробки команди «/help», для цього можна використати подію TELEGRAM_HELP.

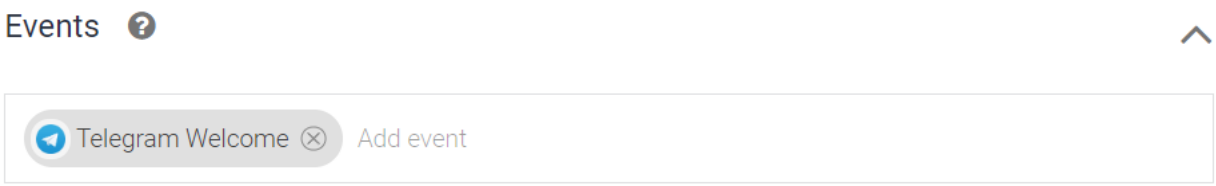

Рис 3.3.1 Створення інтенту у відповідь на команду «/start»

Окрім роботи з командами, необхідно створити інтент, що відповідатиме на питання користувача про функціонал чат-боту. Для цього, можна задати наступні тренувальні фрази:

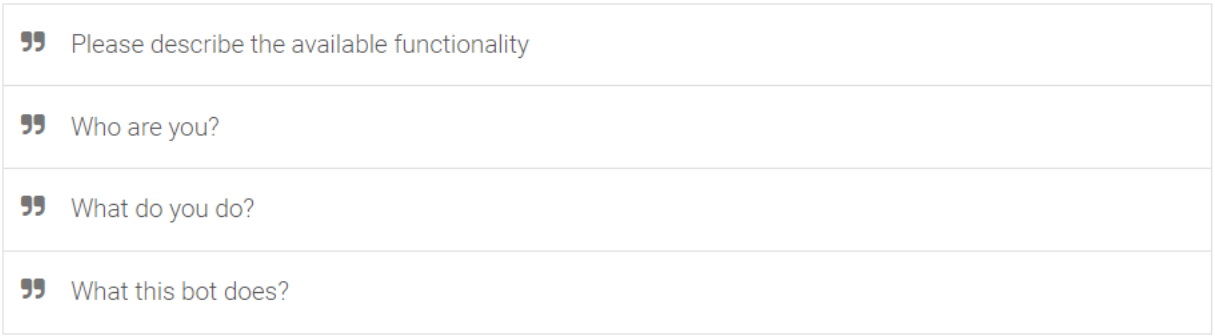

Рис 3.3.2 Тренувальні фрази для створення довідкової інформації

3.4 Використання бази даних Cloud Firestore

Для виконання більшості команд та запитів до чат-боту, є необхідністю збереження даних та їх редагування чи видалення. В рамках цього проекту, було обрано Cloud Firestore – хмарна NoSQL база даних з документо-орієнтованим сховищем. На відміну від баз даних SQL, у Firestore немає таблиць або рядків – замість цього ви зберігаєте дані в документи, які організовуються в колекції для легкої навігації та керування. [16] Використання NoSQL бази даних з гнучкою структурою є виправданим для цього проекту, оскільки це позволить зручно зберігати JSON-об’єкти сутностей, отриманих в Dialogflow. Також, до переваг використання Cloud Firestore можна віднести автоматичне масштабування.

Всього необхідно створити дві колекції – колекцію подій («Events») та колекцію користувачів («Users») для збереження персоніфікованих налаштувань. Для документів, де зберігаються події, було використано наступну структуру:

Рисунок 3.4.1 Приклад збереженої події

Перші чотири поля містять інформацію про подію, наведену в технічному завданні. Ідентифікатор користувача («user_id») зберігається для реалізації запитів на отримання даних про події.

Документ колекції «Users» міститиме лише булеве значення, що вказує чи потрібно щоденно надсилати розклад з метою нагадування. Ідентифікатором кожного з документів колекцій відповідатиме ідентифікатору користувача Telegram.

3.5 Ініціалізація вебхуку

Для реалізації більшості наявного функціоналу, визначеного в технічному завданні, необхідна буде взаємодія з базою даних, реалізованою за допомогою вебхуку. Тому, перед документуванням подальшого процесу розробки, буде доцільним детальніше пояснити принцип роботи вебхуку в рамках цього проекту.

Для його імплементації, було створено Nodejs сервер з використанням веб-фреймворку «express», що має велику кількість допоміжних HTTP-методів та проміжних обробників для розробки серверних застосунків. Також, для парсингу тіла запитів використано бібліотеку «body-parser». Завантаження сторонніх бібліотек здійснюється через менеджер пакетів «npm». Консольна команда для завантаження пакетів має вигляд: *npm install <package_name>*

Код серверу містить наступні частини:

- Ініціалізація express-серверу.
- Підключення до Firestore.
- Розсилка нагадувань про події.
- Обробка POST-запиту для вебхуку.

Код для підключення до Firestore можна скопіювати і вставити в свій застосунок з налаштувань Firestore:

```
const firebaseConfig = {
apiKey: process.env.API_KEY,
authDomain: process.env.AUTH_DOMAIN,
projectId: process.env.PROJECT_ID,
storageBucket: process.env.STORAGE_BUCKET,
messagingSenderId: process.env.MESSANGING_SENDER_ID,
appId: process.env.APP_ID,
```

```
measurementId: process.env.MEASUREMENT_ID,
};

// Initialize Firebase
const firebase = firebaseApp.initializeApp(firebaseConfig);
const db = firestore.getFirestore(firebase);
```

Оскільки в рамках одного чат-боту можна під'єднати лише один вебхук, ідентифікація запиту відбуватиметься за назвою інтенту:

```
//Webhook entry
app.post('/webhook/', async (req, res) => {
if (req.body.queryResult.intent.displayName === "start-cmd")
await eventhandlers.handleStart(db, req, res);
//… other intent handlers
})
```

При розробці та тестуванні, використовувався сервіс serveo для виведення локальних серверів в Інтернет з використанням технології SSH-тунелів. За допомогою нього, можливо перенаправити трафік з HTTPS-домену на локальний сервер. Це значно спрощує розробку, оскільки позбавляє необхідності розгортати кожну зміну коду на сервері. Для цього, використовується наступна команда: *ssh -R chatbotwebhook:80:localhost:1234 serveo.net*

Адреса тестового вебхуку - https://chatbotwebhook.serveo.net/webhook.

3.6 Реалізація створення події у розкладі

Дану задачу можна поділити на два етапи – збір інформації про подію (назва, час початку, тривалість, регулярність) та її збереження у базі даних. В цьому випадку, збір інформації повністю здійснюватиметься за рахунок Dialogflow. Після того, як інформація буде зібрана, відбудеться збереження цих даних.

Для збору інформації, створимо інтент «event-creating» та визначимо його навчальні фрази. Оскільки людина може використати велику кількість синонімів для

події, яку бажатиме зберегти, можна визначити сутність події, що слугуватиме синонімічним рядом. За допомогою визначення синонімів сутності, існуватиме можливість чітко класифікувати дані для майбутнього збереження. Крім цього, в тренувальних фразах, можна використати лише один з цих синонімів, виділивши його як сутність. Це позволяє зберегти час при написанні тренувальних фраз, сфокусувавшись лише на різних варіаціях граматичної структури речення-запиту:

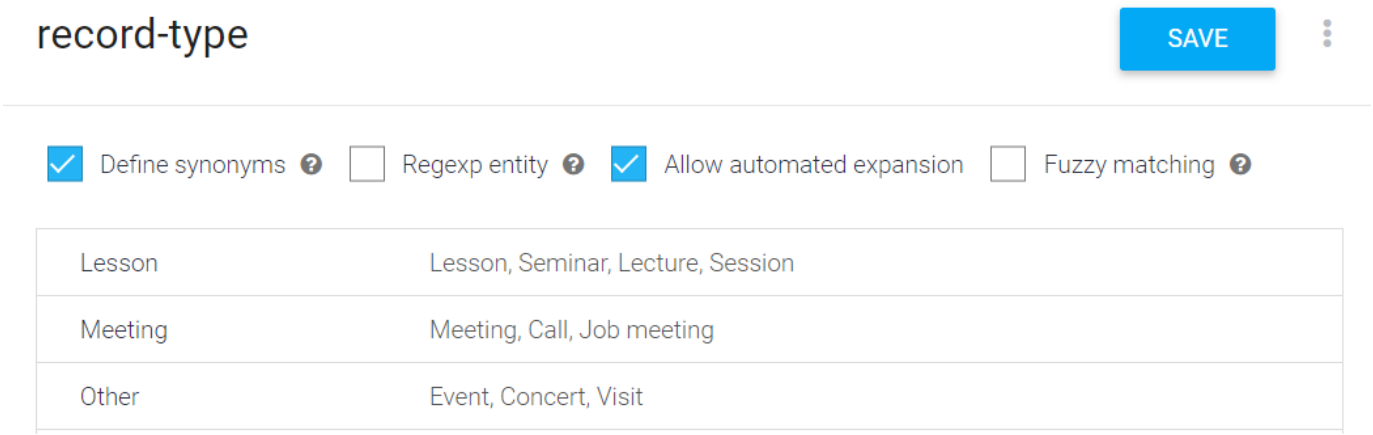

Рис 3.6.1 Сутність типу події

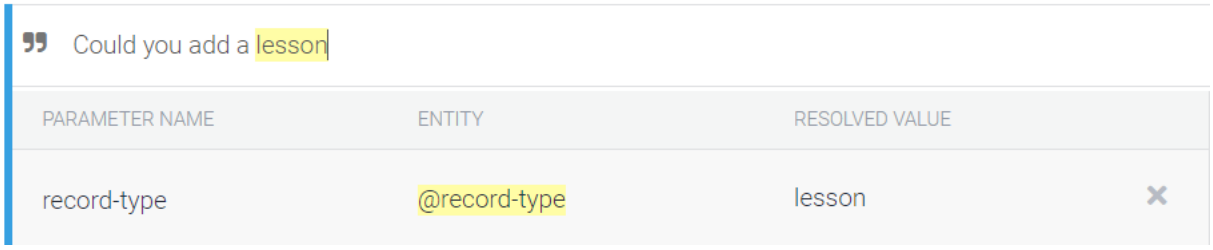

Рис 3.6.2 Розпізнана в тренувальній фразі сутність

Після розпізнання інтенту, бот виводитиме повідомлення про те, яку інформацію буде зібрано. На цьому етапі, можна створити розгалуження діалогу, спитавши у користувача чи готовий він ввести відповідні дані, з використанням follow-up intent. Таким чином, до інтентів, визначених під «event-creating», можна буде перейти лише через повідомлення, яке йде безпосередньо після нього. Інтент «event-creating-no» сигналізуватиме про відмову користувача, а інтент «event-creating-custom» міститиме подальший збір даних, та їх збереження в Firestore:

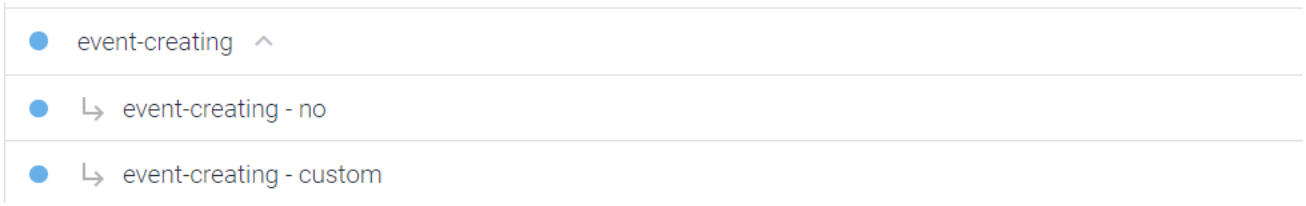

Рис 3.6.3 Приклад розгалуження розмови

Для збору користувацьких даних, використаємо функцію «Дії та параметри» («Actions and parameters»), в якому можна здійснювати почерговий збір даних, визначивши їх типи:

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
☑	event-name	@sys.any	$event-name	☐	Please enter ev...
☑	event-date	@sys.date-ti	$event-date	☐	Please enter wh...
☑	event-duratic	@sys.duratic	$event-duration	☐	Please enter ev...
☑	event-period	@recurring	$event-period	☐	Please enter ho...

Рис 3.6.4 Збір даних про подію

Сутність «recurring» відображає регулярність, з якою відбуватиметься створена користувачем подія. Вона є подібною по структурі до раніше згаданої «record-type», однак використовується не лише для розпізнавання синонімів, а і як параметр події.

В запиті на збереження, будуть передані всі зібрані параметри. Збереження документу в колекцію здійснюється за рахунок функції addDoc:

```
const eventsRef = firestore.collection(db, "Events");
await firestore.addDoc(eventsRef, {
name: name,
user_id: id,
period: period,
timespan: timespan.amount + timespan.unit,
```

```
date: datetime
});
```

В результаті успішного виконання, в відповідь на цей та інші POST-запити, буде додано відповідне текстове повідомлення.

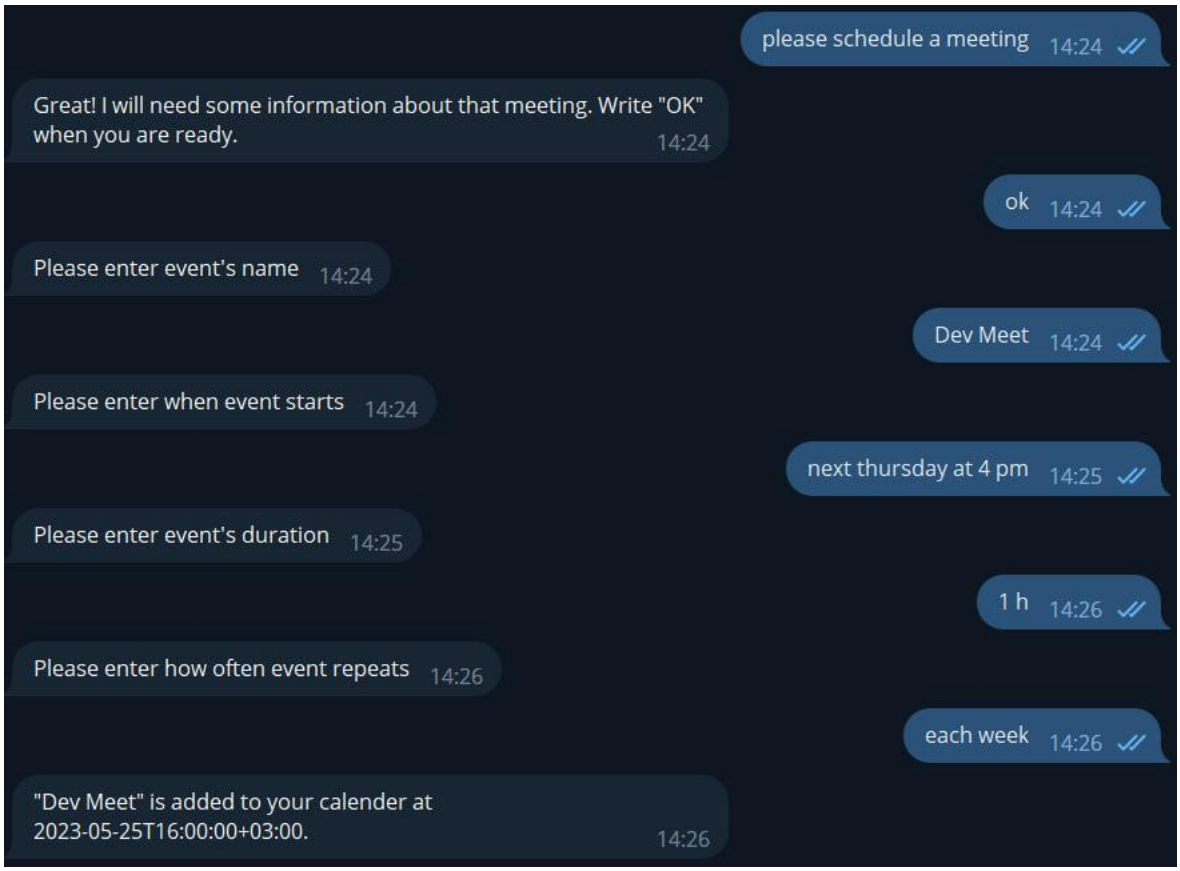

Рис 3.6.5 Створення події у розкладі

3.7 Запити на отримання розкладу подій

Для обробки запитів на отримання розкладу, створено наступні інтенти з префіксом «view»:

- view-day – перегляд за вказаною користувачем датою (сутність «@sys.date»)
- view-today-cmd – перегляд сьогоднішніх подій, з використанням команди «/day»

- view-week – перегляд подій наступних 7 днів, починаючи з сьогоднішнього дня.
- view-week-cmd – аналог попередньої функції, працює через команду «/week».

Розглянемо роботу цих запитів на прикладі запиту по конкретній даті (інтент «view-day»), функція «handleViewDateSchedule». Для початку, необхідно здійснити запит на отримання подій.

```
let id = req.body.originalDetectIntentRequest.payload.data.from.id;
const q = firestore.query(firestore.collection(db, "Events"), firestore.where("user_id",
"==", id));
const events = await firestore.getDocs(q);
```

Далі, з запиту дістається дата, на яку необхідно отримати розклад, та здійснюється фільтрування по всім подіям користувача. Оскільки необхідно врахувати регулярність, замість доповнення запиту на Firestore фільтром даті проведення події, використовується власна логіка фільтрування, що опрацьовує всі події конкретного користувача – *filterScheduleByDay*. Для побудови повідомлення з вказанням та сортуванню по днях тижня, використовується функція *composeSchedule*, яка отримує відфільтровані по датах події:

```
let queryDate = req.body.queryResult.outputContexts[0].parameters.date || new
Date().toISOString()
let eventByDate = util.filterScheduleByDay(events, queryDate);
let message = util.composeSchedule(eventByDate);
res.send(util.stringToMsgObject(message));
},
```

Запити, реалізовані в інших інтентах, працюють за тим самим принципом, але на вхід отримують сьогоднішню дату чи дати всіх днів цього тижня.

Рис 3.7.1 Запит на розклад по дню

3.8 Нагадування про події

Важливою складовою застосунку календаря, в тому числі реалізованого через чат-бота, є нагадування, що допомагають користувачам не пропустити події. Для їх імплементації, застосовано бібліотеку «node-cron», що позволяє планувати автоматичне виконання скриптів в заданий час. В ній використовується синтаксис, подібний до cron-утиліт в Unix-системах:

```
# ┌────────────── second (optional)
# │ ┌──────────── minute
# │ │ ┌────────── hour
# │ │ │ ┌──────── day of month
# │ │ │ │ ┌────── month
# │ │ │ │ │ ┌──── day of week
# │ │ │ │ │ │
# │ │ │ │ │ │
# * * * * * *
```

Рис 3.8.1 Схема синтаксису cron-команд [17]

Код налаштування щоденних сповіщень в 00:00 матиме наступний вигляд:

```
cron.schedule('0 0 * * *', async function () {//Код функції}, {//Параметри});
```

Для підключення до Telegram API та розсилки повідомлень, використовуватиметься бібліотека «telegraf». Отримання розкладу здійснюється тими ж функціями, що застосовувались для вищезгаданих інтентів для запиту:

```
//Підключення до API
const bot = new Telegraf(process.env.BOT_TOKEN);
//Відправка повідомлення
await bot.telegram.sendMessage(user.id, scheduleMsg);
```

В деяких випадках, кінцевий користувач може захотіти відключити нагадування, дана функція реалізована інтентом «notification-toggle». Такий інтент є частковим випадком, де користувач може вказати різні дії. Для їх розпізнавання, можна використати сутність, що вказує на дію ввімкнення або вимкнення. В залежності від отриманої дії, збережеться відповідне налаштування сповіщень в колекції «Users».

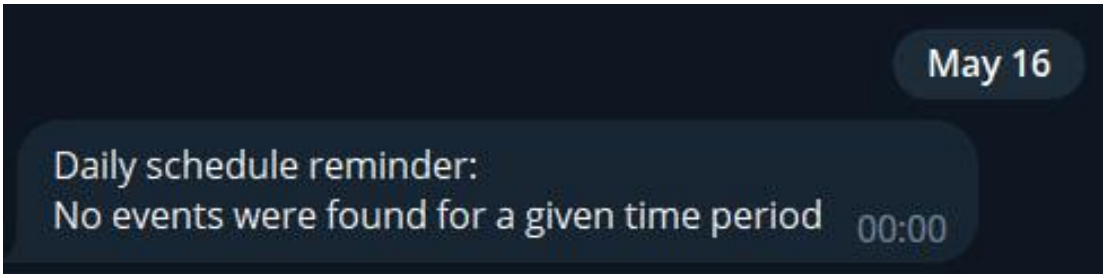

Рис 3.8.2 Щоденне нагадування про події

3.9 Дослідження впливу різноманітних факторів на розпізнання наміру

В попередньому розділі, була згадана можливість налаштування порогу класифікації. Маючи визначені інтенти, можливо провести дослідження впливу цього порогу на правильність розпізнання намірів. Для цього, створимо текстовий файл з двадцятьма можливими варіантами користувацького вводу. Десять з них за своєю суттю відповідатимуть інтенту створення події «event-creating», п’ять – інтенту видалення події «event-deletion», та ще п’ять – не відповідатимуть жодному з інтентів та міститимуть запити про обробку подій в JavaScript. Дані запити мають схожість з передбаченими через використання слова «event» в обох випадках. Текст запитів наведено в додатку до роботи.

Для порівняння використаємо три значення порогу класифікації – низьке (0.1), значення за замовчуванням (0,3) та високе (0,9). Результат тренування з всіма трьома значеннями показав схожий результат:

- 2 речення, що за суттю відповідають створенню подій, не були розпізнані через використання дієслова «plan», не передбаченого тренувальними фразами.
- 4 з 5 запитів про обробку подій в JavaScript були визначені як хибно позитивні. Також, в 4 з 5 цих запитів слово «event» було помилково виділено як сутність, хоча в цьому випадку даний результат є хибним, бо не підходить по контексту.

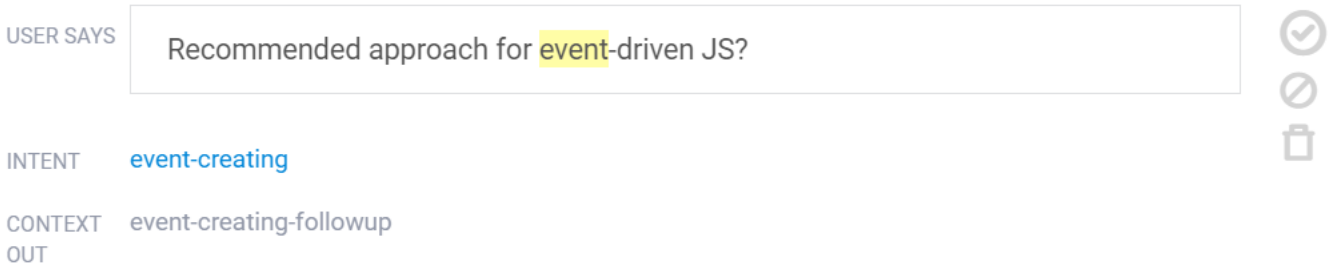

Рис 3.9.1 Помилка в розпізнанні сутності

Для покращення результатів, додамо запитання про JavaScript в резервний інтент, щоб зробити їх негативними прикладами. Навіть після заміни цих запитів на нові питання про обробку подій в JavaScript, нові запити не будуть розпізнані як хибно позитивні. При заміні 5 запитів про обробку подій в JavaScript на запити з пошуком або рекомендацією книжок, що не містять схожих слів з визначеними в боті інтентами, усі ці запити не будуть розпізнані, без додавання до негативних прикладів.

Також, протестуємо максимально подібні за структурою і текстом речення, але з використанням різних за значенням іменників. Повідомлення «please add a meet (з англ. зустріч)» буде розпізнано як намір створення події, в той час як «please add a meat (з англ. м’ясо)» не буде розпізнано, хоча різниця між ними – лише одна літера.

На основі проведеного досліду, можемо зробити наступні висновки:

- Поріг класифікації не є найважливішим чинником при розпізнаванні інтентів.
- Пріоритетним в розпізнаванні є значення іменників та дієслів, а не подібність синтаксичних структур чи подібність в прямому порівнянні.
- Іменники та дієслова є найважливішими факторами для розпізнавання наміру.
- Використання негативних прикладів є ефективним способом обробки крайніх випадків, де однакове слово може бути вживаним в різних контекстах.

Висновок до розділу 3

Задукоментовано процес розробки чат-боту - календаря з використанням платформи Dialogflow. Детально описані різноманітні можливості використання функціоналу інтентів та сутностей. Описано принцип інтеграції чат-боту з базою даних Firestore через вебхук. Проведено дослідження роботи алгоритмів РПМ, що використовуються в Dialogflow.

Загальний висновок до кваліфікаційної роботи

У цій роботі було детально описано предметну область чат-ботів та роль технологій штучного інтелекту в їх розвитку. Наведено класифікацію чат-ботів за принципами їх роботи: спосіб вводу, використання розуміння природної мови (РПМ) чи правил, орієнтованість на завдання. Також описані сфери застосування чат-ботів: бізнес, розваги, утиліти та анкетування. На основі практичних кейсів, наведено порівняння випадків застосування чат-ботів з їхніми альтернативами, перелічені основні переваги застосування чат-ботів.

Детально описано РПМ-платформу Dialogflow ES, наведено її основні переваги. Пояснені основні концепти, що застосовуються для розробки чат-ботів в цій платформі, а саме:

- Агенти
- Інтенти
- Сутності
- Виконання стороннього коду
- Використання машинного навчання

Пояснена архітектура чат-ботів на основі інтентів, вказано її переваги та недоліки.

З використанням Dialogflow, розроблено власного чат-бота з можливістю розуміння природної мови в месенджері Telegram. Описано процес роботи з вебхуком на Nodejs, збереження та відображання користувацьких даних в БД Firestore. Наведено приклад планування виконання завдань з синтаксисом «cron». Обґрунтовано доцільність використання всіх вищезгаданих технологій. Проведено дослідження алгоритмів РПМ платформи Dialogflow. Досліджено вплив порогу класифікації на розуміння користувацьких повідомлень, доведено ефективність застосування негативних прикладів для покращення точності розуміння контексту.

Список літератури

1. Вікімедіа, У. П. (2022). Чат-бот. uk.wikipedia.org. https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82-%D0%B1%D0%BE%D1%82
2. Електронне джерело: https://openai.com/research/openai-five-defeats-dota-2-world-champions
3. Mason, J. (n.d.). Understanding English in limited pragmatic domains. http://www.yorku.ca/jmason/UnderstandingEnglishInLimitedPragmaticDomains.html
4. A Survey on Conversational Agents/ChatbotsClassification and Design TechniquesShafquat Hussain(&), Omid Ameri Sianaki, and Nedal Ababneh
5. A Survey on Conversational Agents/ChatbotsClassification and Design TechniquesShafquat Hussain(&), Omid Ameri Sianaki, and Nedal Ababneh
6. Wikipedia contributors. (2022, December 25). Dialogflow. Wikipedia. https://en.wikipedia.org/wiki/Dialogflow
7. Dialogflow release notes. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/docs/release-notes#November_16_2017
8. Dialogflow CX: Build a retail virtual agent | Google Codelabs. (n.d.). Google Codelabs. https://codelabs.developers.google.com/codelabs/dialogflow-cx-retail-agent#0
9. Agents. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/agents-overview
10. Intents. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/intents-overview
11. Default intents. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/intents-default
12. Entities. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/entities-overview

13. Вікімедіа, У. П. (2022). Webhook. uk.wikipedia.org. https://uk.wikipedia.org/wiki/Webhook
14. Intent matching. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/intents-matching#confidence
15. Training. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/es/docs/training
16. Stape. (2020). Firestore із server Google Tag Manager - Stape. stape.io. https://stape.io/ua/blog/zapis-dannih-v-firestore-s-server-google-tag-manager-ua
17. npm: node-cron. (n.d.). Npm. https://www.npmjs.com/package/node-cron