

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ЧАТБОТА ОБРОБКИ ПРИРОДНОЇ МОВИ.
DEVELOPMENT OF A CHATBOT TO PROCESS NATURAL LANGUAGE

Виконав: студент(ка) 4 курсу, групи КНТ-218

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Гадун Е.В.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гадун Еліні Віталійовні
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка чатбота обробки природної мови.
Development of a Chatbot to Process Natural Language

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань .
2. Технології і методи розробки чатботів. 3. Опис програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю. доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання « 04 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис)

Гадун Е.В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 64 с., 29 рис., 3 табл., 4 дод., 19 джерел.

ЧАТБОТ, ШТУЧНИЙ ІНТЕЛЕКТ, МОДЕЛЬ, ФАКТИ, ТЕКСТ, ПОВІДОМЛЕННЯ, ПРИРОДНА МОВА, РОЗМОВА, PYTHON, TELEGRAM, TENSORFLOW, KERNES, API.

Об'єкт дослідження – програмні засоби для онлайн спілкування природною мовою.

Предмет дослідження – чатбот обробки природної мови.

Мета роботи – створення чатботу, який спілкується та розуміє природну мову співрозмовника.

Матеріали, методи та технічні засоби: евристична модель побудови чатботу. Для реалізації роботи використано Python 3.7. Чатбот завантажений на хмарний сервіс “Heroku”. Для створення чатботів у Telegram використана бібліотека python-telegram-bot. Навчання моделі ШІ проводилось за допомогою бібліотек tensorFow та Kernes.

Результати. Створено чатбот, який імітує реального співрозмовника та підтримує бесіду.

Висновки. Розроблений чатбот можна використовувати в онлайн підтримки, сервісних центрах, в допомогу роботі менеджера.

Галузь використання – онлайн сервіси підтримки.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 64 pages, 41 figures, 29 tables, 4 appendixes, 19 sources.

CHATBOT, ARTIFICIAL INTELLIGENCE, MODEL, FACTS, TEXT, MESSAGE, NATURAL LANGUAGE, CONVERSATION, PYTHON, TELEGRAM, TENSORFLOW, KERNES, API.

Object of study is a software for online communication in natural language..

Subject of study are chatbot natural language processing.

The purpose of the work is to create a chatbot that communicates and understands the natural language of the interlocutor..

Materials, methods, and tools: heuristic model of chatbot construction. Python 3.7 is used to implement the robot. The chatbot is uploaded to the Heroku cloud service. The Telegram library uses the python-telegram-bot library to create chatbots. The AI model was taught using the tensorFow and Kernes libraries.

Results. A chatbot has been created that simulates a real interlocutor and supports the conversation.

Conclusions The developed chatbot can be used in online support, service centers, to help the manager.

The field of use is a online support services.

ЗМІСТ

С.

ВСТУП	8
1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ	10
1.1 Огляд чатботів	10
1.2 Чатбот платформи	17
1.3 Чатботи обробки природної мови	18
1.4 Висновки за розділом.....	24
2 ТЕХНОЛОГІЇ І МЕТОДИ РОЗРОБКИ ЧАТБОТІВ	26
2.1 Технології розробки чатботів.....	26
2.2 NLP-технології	27
2.4 Проєктування чатботу	30
2.4 Висновки за розділом.....	33
3 ОПИС ПРОГРАМИ.....	34
3.1 Попередня обробка тексту	34
3.2 Розробка чатботу	36
3.3 Висновки за розділом.....	41
4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ	42
4.1 Реєстрація чатботу	42
4.2 Тестування чатботу	43
4.4 Висновки за розділом.....	45
ВИСНОВКИ.....	46
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	47
Додаток А Текст програми.....	49
Додаток Б Приклади фактів	52
Додаток В Текст програми.....	54
Додаток Г Слайди презентації	57

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AIML	– Artificial Intelligence Markup Language;
API	– Application Programming Interface;
IDE	– Integrated Development Environment,
NLP	– Natural Language Programming;
NLU	– Natural Language Understanding;
БД	– База даних;
БЗ	– База знання;
РЧ	– Розмовний чатбот;
ШІ	– Штучний інтелект.

ВСТУП

Сучасні засоби спілкування в мережі стало частиною життя кожної людини. Знаходитись онлайн можливо через мережі інтернет та мобільний зв'язок. За допомогою месенджерів можливо отримувати різноманітну інформацію. Вони допомагають при роботі та навчанні, чи просто залишатись на зв'язку з рідними.

Мета проєкту – це розробка чатботу обробки природної мови. Актуальність обраної теми складається з того, що віртуальне спілкування є дуже значимі для людини. В кожного користувача інтернетом є більше двох видів додатків для комунікації. Таким чином, сучасні клієнтоорієнтовані інформаційні системи все частіше мають у своєму складі чатботи в різних месенджерах для мовного спілкування та обміну інформацією.

Популярність чатботів - це не просто віяння якихось сучасних трендів, а дійсно дуже зручна та ефективна річ для вирішення типових завдань, наприклад: допомогти користувачеві зробити замовлення; прийняти заявку; зібрати зворотний зв'язок від користувачів; дати консультацію; зібрати контактні дані; реалізувати різні товари та послуги; завантажити музику, фільм чи книгу.

Розмовні чатбот – це програма на основі штучного інтелекту (ШІ), яка імітує людські розмови. Їх ще називають цифровими помічниками, котрі розуміють людські запити. Боти інтерпретують та обробляють запити користувачів та швидко дають відповідні відповіді.

Боти можуть використовувати як голос, так і текст, можуть бути розміщені на веб-сайтах, додатках і месенджерах, таких як Facebook Messenger, Telegram, Whatsapp і т.п.

РЧ аналізує запит користувача, щоб дати найбільш релевантну відповідь на цей запит – це є його ключовим завданням. Після завершення аналізу користувач отримує відповідь на запит.

Виходячи з поставленої мети, було визначено такі завдання розробки:

- провести аналіз сучасних чатботів та платформ для їх реалізації;
- обрати технології для розробки чатбота обробки природної мови;
- розробити розмовний чатбот;
- провести тестування чатботу.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Огляд чатботів

Чатботи – це програми-співрозмовники, які імітують реальний усний чи письмовий діалог із користувачем. Автоматизацію спілкування з клієнтами останніми роками активно використовує бізнес. Чатботи інтегруються з системами керування вмістом, завдяки чому відбувається автоматичне розсилання опитувань, повідомлень про статус замовлення і т. д. [1]

Для початку варто навести одну фундаментальну річ: чатбот - це, перш за все, додаток, який нічим не поступається нативним додатком або сайту. В основі всіх згаданих вище рішень зазвичай лежить інтерфейс користувача (UI, Front-End) і якийсь набір дій з боку сервера, який обробляє запити користувача (Server, Back-End), а також взаємодіє з базою даних та іншими сервісами, необхідними для реалізації вимог до товару. Це стандартна класифікація [2].

По суті, чатбот - це такий же продукт, як і сайт, але він має одну важливу особливість: робота над інтерфейсом користувача віддається в треті руки. Тобто ми реалізовуємо логіку роботи програми, а відображення та взаємодію проводимо через інтерфейс чату (природно, у сучасному розумінні це якийсь месенджер). На даному етапі варто відкинути питання побудови цього самого інтерфейсу для зручності використання через третю програму (славнозвісний UX), а також сам процес адаптації представлення вашого додатка до інтерфейсу та можливостей чат-додатка [3].

Враховуючи відповідь на питання, що таке чатбот, ми можемо зробити висновок, що він може вирішувати той же спектр завдань, що будь-який інший продукт. Переваги та недоліки сучасних чатботів наведені в таблиці 1.1.

Таблиця 1.1 – Аналіз сучасних чатботів

Особливості чатботів	- найчастіше мінімальна можливість взаємодії з файловою системою та операційною системою користувача; - обмеженість інтерфейсу вибраним чат-додатком; - залежність від працездатності зовнішнього продукту (це дуже актуально останнім часом); - максимально утруднене автоматизоване тестування.
Переваги	- користувач знаходиться в знайомому та зрозумілому для нього інтерфейсі; - можливість масштабування.
Недоліки	- це конструювання зручного та функціонального інтерфейсу вже на базі існуючого; - різноманітне отримання складного сценарію поведінки користувача в багатьох месенджерах.

Звичайно, враховуючи деякі особливості:

- найчастіше мінімальна можливість взаємодії з файловою системою та операційною системою користувача (на відміну від нативної програми);
- обмеженість інтерфейсу вибраним чат-додатком;
- залежність від працездатності зовнішнього продукту (це дуже актуально останнім часом);
- максимально утруднене автоматизоване тестування.

Основна перевага чатбота в тому, що користувач знаходиться в знайомому та зрозумілому для нього інтерфейсі та взаємодіє з нашим додатком, не втрачаючи фокусу на спілкуванні. Особливо це важливо для роботів, які вирішують завдання всередині компанії/команди [4].

Звичайно ж, найбільша корисність у готовому інтерфейсі - це те, що у всіх, хто користується вашим додатком, при правильній та логічній архітектурі, підсумковий результат виглядатиме однаково та максимально

передбачувано. Маленьким плюсом можна вважати також можливість масштабування за рахунок підняття нових ідентичних роботів, за належної архітектури на одному сервері.

Основна складність у створенні програми під месенджери - це конструювання зручного та функціонального інтерфейсу, виходячи з того, які особливості є у нас у розпорядженні всередині месенджера. Відповідно, створюючи певні user-flow, , з якими варто змиритися, а іноді, як у випадку зі Slack-ом, - величезні можливості, виходячи з тих блоків, які нам дозволено використовувати [5].

Якщо потрібно створити складну послідовність дій і реалізувати якийсь більш-менш виразний спосіб отримання складного сценарію поведінки користувача в багатьох месенджерах, ми зіткнемося з великими проблемами щодо зручності побудови відповідного інтерфейсу. Наприклад, це може бути складна форма, в якій послідовність заповнення/вибору впливає на наступний крок та/або на дії всередині програми [6].

Сучасні чатботи займають велике місце в реальному житті. Звичайно, насамперед варто згадати ботів для криптовалюти та роботи з нею, покупки авіа- та залізничних квитків. Так, можна знайти багато прикладів із подібним функціоналом. Наприклад, державна залізнична компанія “Укрзалізниця” має боти з продажу квитків у Telegram та Viber.

Великий сегмент ботів закриває завдання взаємодії всередині месенджера. Це як офіційні представництва та чати для спілкування з відомими людьми чи компаніями, так і додатковий функціонал для модерації різноманітних чатів та спільнот. Наступною за списком можна назвати групу, у якій ботів можна охарактеризувати як помічників у повсякденній діяльності користувачів. Це як боти для обліку фінансів, так і різноманітних допоміжні інструменти для моніторингу та управління різними ресурсами, не виходячи з улюбленого месенджера.

Крім того, у цю нішу поступово приходить і великий бізнес. Багато компаній організовують роботу своєї технічної підтримки через ботів і

дозволяють при цьому робити кроссплатформенну взаємодію. Це означає, що люди можуть написати такому боту з різних месенджерів, але при цьому мати схоже або ідентичне виконання будь-якого запиту.

Варто також згадати, що багато робот, які зараз можна знайти в тому ж Telegram і які мають порівняно велику кількість користувачів, які мають додатковий функціонал, доступний за рахунок донату: як у формі передплати, так і одноразовим платежем. Такі боти найчастіше комерційно успішні, оскільки зміст сервера для роботи бота коштує в рази менше, ніж сайту зі схожим навантаженням за рахунок розподілу затребуваних користувачем ресурсів, у тому числі на сервері месенджера [7].

У чатботів є кілька класифікацій: за алгоритмом, видом та функціональністю. Класифікація чатботів наведена в таблиці 1.2.

Таблиця 1.2 – Класифікація чатботів

Критерій	Класифікація
За алгоритмом	- примітивні; - чатботи.
На вигляд	- кнопкові; - текстові.
За функціоналом	- комунікаційні; - функціональні.

За алгоритмом бувають примітивні і чатботи, що саморозвиваються [7].

Примітивні. До примітивних (обмежених) відносяться найпростіші чатботи. Вони мають невеликий функціонал і заздалегідь підготовлений скрипт, яким вони спілкуються з користувачами.

Саморозвиваються. В основі чатбота, що саморозвивається, лежить нейронна мережа. Завдяки цьому бот розуміє, що пише користувач і,

аналізуючи інформацію, видає цілком реалістичні відповіді. Іноді може здатися, що ви спілкуєтесь зі справжньою людиною!

На вигляд чатботи діляться на кнопкові та текстові [4].

Кнопковий. Спілкування із чатботом відбувається за допомогою спеціальних кнопок, на яких запропоновані заздалегідь підготовлені відповіді.

Текстовий. Діалог між чатботом та користувачем відбувається за допомогою тексту. Бот розпізнає ключові слова у запиті користувача та на основі отриманої інформації видає відповідь.

За функціоналом виділяють комунікаційних та функціональних чатботів.

Комунікаційні. Комунікаційні чатботи беруть він обов'язки консультантів. Їхнє основне завдання - це спілкування з користувачем. У їх функціонал може входити відповіді на запитання користувачів, допомога у підборі товару або послуги, а також інформування про різні акції та знижки.

Функціональні. Завдяки функціональним чатботам ставати можливим здійснювати певні дії, такі як купівля товарів чи послуг, проведення оплати тощо.

Популярність чатботів - це не просто віяння якихось сучасних трендів, а дійсно дуже зручна та ефективна річ для вирішення типових завдань, наприклад [2]:

- допомогти користувачеві зробити замовлення;
- прийняти заявку;
- зібрати зворотний зв'язок від користувачів;
- дати консультацію;
- зібрати контактні дані;
- реалізувати різні товари та послуги;
- допомогти користувачеві записатися в салон краси/ прийом до лікаря та ін;
- завантажити музику, фільм чи книгу.

Розмовні чатбот – це програма на основі штучного інтелекту (ШІ), яка імітує людські розмови. Їх ще називають цифровими помічниками, котрі розуміють людські запити. Боти інтерпретують та обробляють запити користувачів та швидко дають відповідні відповіді [8].

Боти можуть використовувати як голос, так і текст, можуть бути розміщені на веб-сайтах, додатках і месенджерах, таких як Facebook Messenger, Telegram, Whatsapp і т.п.

РЧ аналізує запит користувача, щоб дати найбільш релевантну відповідь на цей запит – це є його ключовим завданням. Після завершення аналізу користувач отримує відповідь на запит. РЧ працюють із використанням трьох методів класифікації:

а) зіставлення з шаблоном. Боти використовують збіги шаблонів для групування тексту, щоб вибрати відповідну відповідь. Мова розмітки штучного інтелекту (AIML) – це стандартна структурована модель цих шаблонів. Бот може отримати правильну відповідь, порівнявши запит із шаблоном. Таке зіставлення вимагає багато попередньо згенерованих шаблонів.

б) розуміння природної мови (NLU). NLU (Natural language understanding, розуміння природної мови) - це здатність РЧ розуміти людину. Це процес перетворення тексту на структуровані дані для розуміння машиною (рис. 1.1.). NLU слідує трьом конкретним концепціям: сутності, контекст та очікування [3].

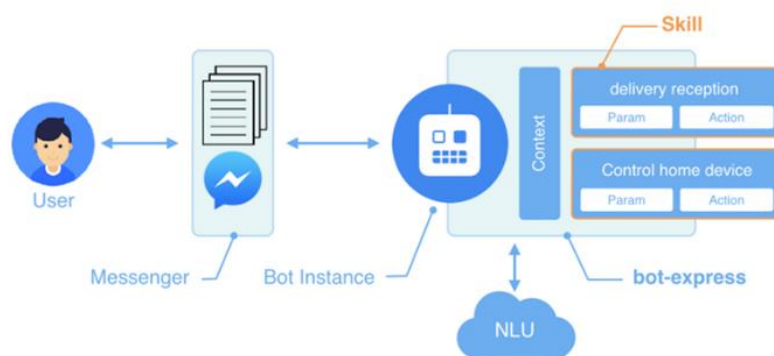


Рисунок 1.1 - РЧ розуміння природної мови NLU [3]

1) сутність - сама ідея РЧ. Наприклад, це може бути система повернення грошей у електронної комерції;

2) контекст є алгоритмом розпізнавання природної мови, який ідентифікує запит. Контекст запиту підкаже боту відповідь;

3) очікування — бот повинен мати можливість виправдати очікування клієнтів, коли вони вводять запит.

в) обробка природної мови (NLP). Боти для обробки природної мови (NLP) призначені для перетворення тексту або мови користувача в структуровані дані (рис. 1.2). Дані надалі використовуються для вибору відповідної відповіді [8].

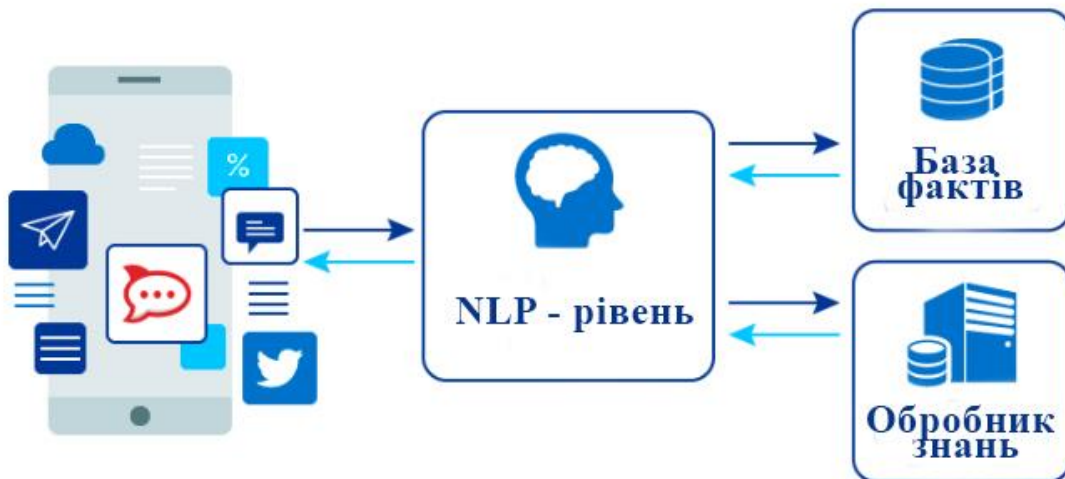


Рисунок 1.2 - Обробка природної мови NLP

Обробка природної мови (NLP) складається з наступних кроків:

- токенизація – NLP фільтрує набір слів у формі токенів;
- аналіз настрою – бот інтерпретує відповіді користувачів відповідно до їх емоцій;
- нормалізація – перевіряє патерни (вони ж шаблони), які можуть змінити сенс запиту користувача;

- розпізнавання сутностей – робота шукає різні категорії необхідної інформації.
- аналіз залежностей – чатбот шукає найпоширеніші фрази, які можуть використовувати користувачі.

1.2 Чатбот платформи

Згідно з дослідженням компанії We Are Social, середньостатистичний міський житель віком від 16 до 64 років проводить в Інтернеті майже 7 годин. Причому, у 67% випадків заходить до мережі з мобільного телефону [6].

За даними Online Market Intelligence (OMI) та Content Research у 83% власників смартфонів є один або кілька месенджерів. У п'ятірку провідних у Україні з використання входять: WhatsApp, Viber, Telegram, Skype (рис. 1.3) [4].

Мобільний застосунок	%
Viber (Вайбер чи вібер)	73,6
WhatsApp (Вотсап)	25,3
Месенджер Фейсбуку	42,7
Телеграм	31,6
Твіттер	4,9
Сігнал	3,8
Скайп	11,3
Інстаграм	26,4
Тік Ток	9,1
Інше (ЗАПИШІТЬ)_____	2,6
Нічим з перерахованого	18,7
ВАЖКО СКАЗАТИ/ ВІДМОВА ВІДПОВІДАТИ	0,8

Рисунок 1.3 – Популярність месенджерів в Україні

По-перше, створювати віртуального помічника потрібно у тому месенджері, де знаходиться цільова аудиторія. По-друге, функціонал

подібного програмного забезпечення в WhatsApp дуже обмежений, і сам месенджер не поспішає давати можливість впроваджувати чатботів.

Що стосується решти месенджерів - найпросунутішим і дружнім по відношенню до чатботів вважається Telegram. Це один із перших майданчиків, який надав можливість їх створювати.

Facebook Messenger. Одна з найпопулярніших платформ для чатботів у світі налічує понад сто тисяч активних видів. Має великий набір функцій: шаблони списків товарів, розширення, чеки, шаблони реєстрації рейс, аутентифікацію, прийом платежів тощо.

Viber. Публічні облікові записи для бізнесу з'явилися ще в 2016 році, можна налаштувати спілкування з передплатниками облікового запису через бот або CRM.

Що стосується продажів в Instagram, то поки що немає легальної можливості для запуску ботів [9].

1.3 Чатботи обробки природної мови

Чатбот може генерувати відповіді з нуля на основі моделей машинного навчання або використовувати деякі евристики для вибору деяких відповідей із зарезервованої бази даних.

Однак генеративні моделі дуже складно побудувати та навчити. Зазвичай потрібні мільйони прикладів, щоб створити генеративну модель, щоб отримати більш якісний діалог, і велика вада генеративних моделей полягає в тому, що ви не можете бути повністю впевнені в тому, яка відповідь модель викличе. Microsoft Tay потрапляє до цієї категорії [10].

Генеративні моделі, безперечно, є тенденцією розвитку чатботів у майбутньому (рис. 1.4). Вони можуть зробити роботів розумнішими. Але цей метод не набув широкого поширення розробниками чатботів, він більше використовується в лабораторії.

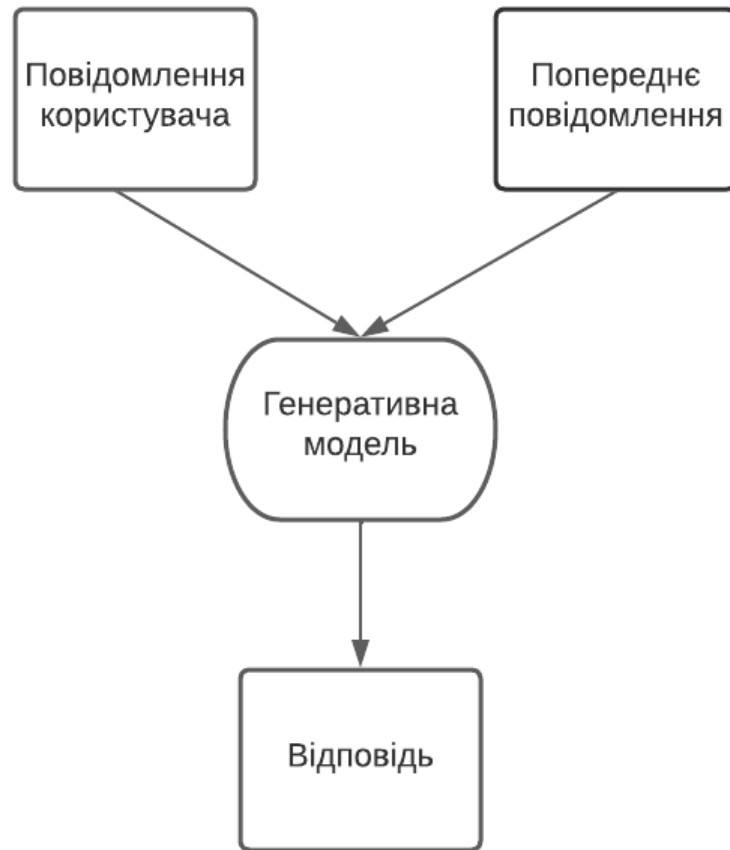


Рисунок 1.4 – Генеративна модель

Модель на основі пошуку. У порівнянні з генеративними моделями, розробка моделі пошуку дуже проста. Модель надасть більше можливих результатів прогнозу. Якщо модель не може отримати відповідь зі 100% точністю, модель поверне найімовірнішу відповідь і переконайтеся, що немає неправильних або граматичних помилок [11].

В даний час моделі на основі пошуку більш практичні, і розробники можуть використовувати безліч алгоритмів та API (рис. 1.5).

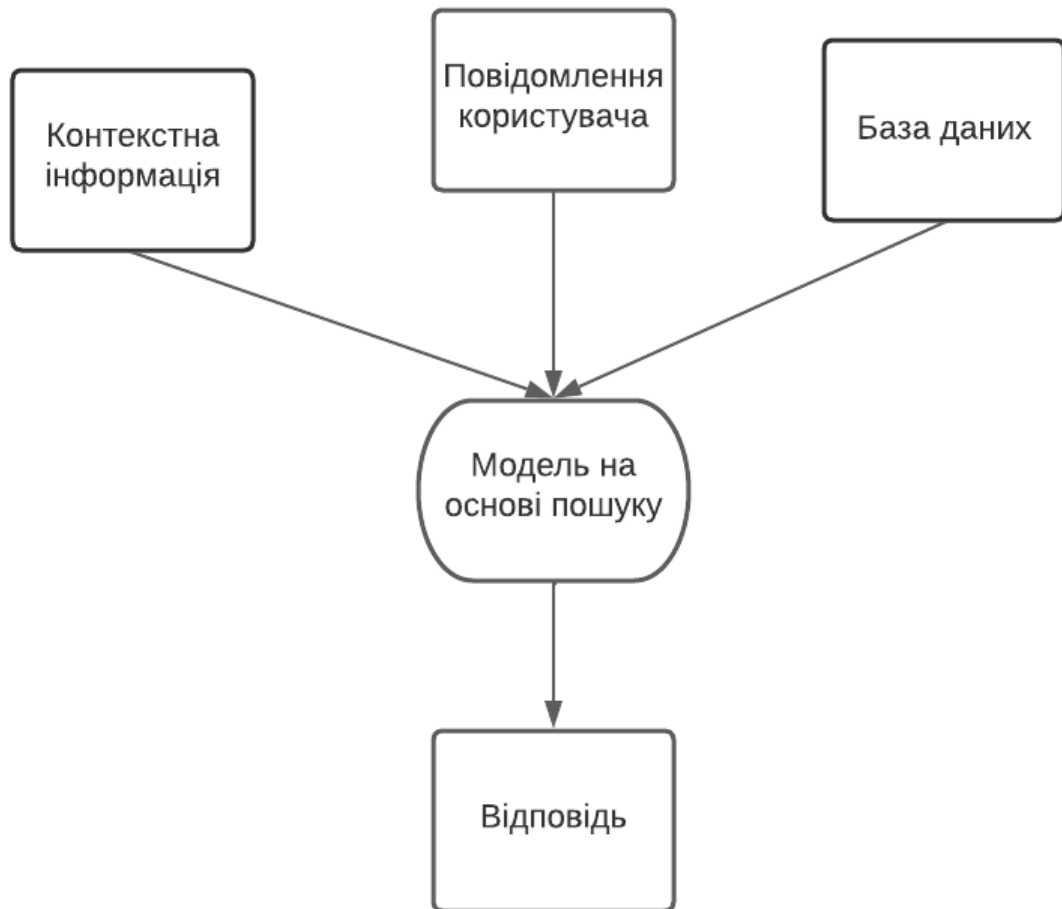


Рисунок 1.5 – Модель на основі пошуку

Чатбот вибирає найкращу відповідь з бази даних кандидатів на основі введення користувача та контекстної інформації. Контекстна інформація може включати поточну інформацію в діалоговому вікні, інформацію про попереднє діалогове вікно і змінну інформацію, збережену в історії (наприклад, деякі налаштування користувача).

Якщо робот не використовує контекстну інформацію, це зв'язок без збереження стану. Він буде реагувати лише на поточну інформацію користувача та не оброблятиме історичну інформацію.

Евристична модель на основі зіставлення із зразком. Для розробки моделей евристичного зіставлення можливо багато різних методів проектування, таких як умовна логіка if-else, методи класифікації машинного

навчання і т. д. Найпростіший спосіб - використовувати набір правил, де шаблон - це умова правила. Ця модель дуже популярна для створення розважальних чатботів. AIML Це мова для написання шаблонів та відповідей (рис. 1.6) [12].

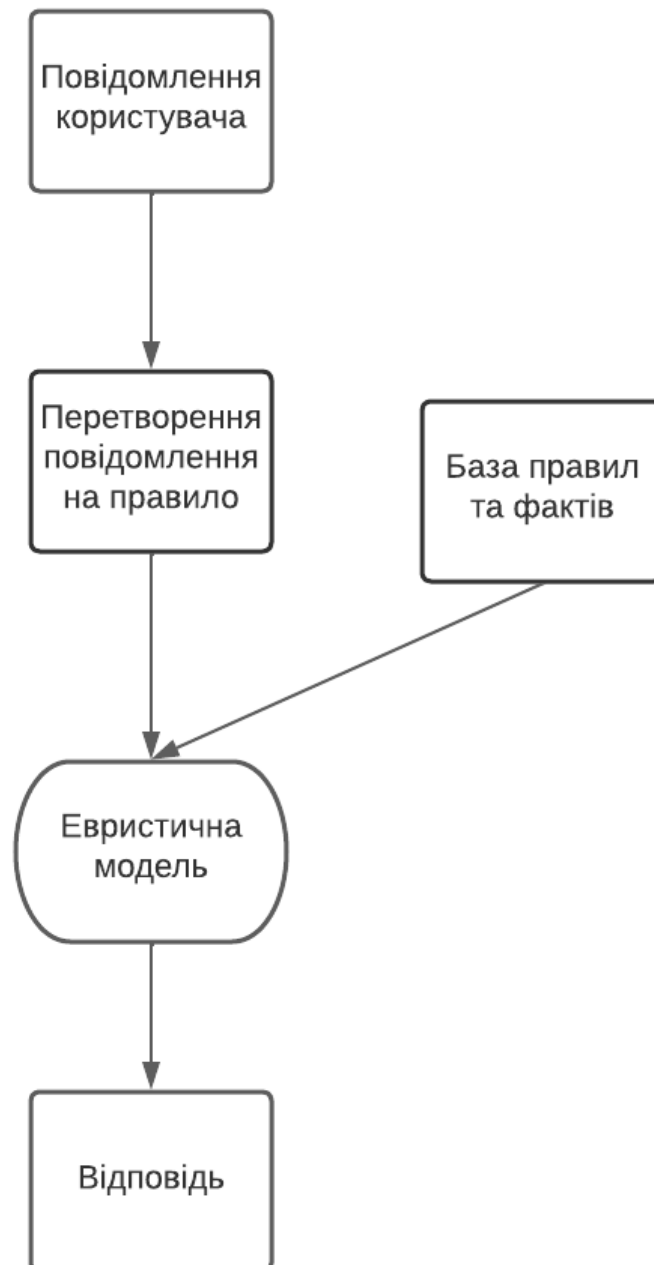


Рисунок 1.6 – Евристична модель

Коли робот отримує інформацію користувача, він спрощує всі шаблони, доки знайде збіг з введенням користувача. Потім AIML згенерує відповідну відповідь на основі знайденого шаблону.

ChatScript Це подальша реалізація цієї ідеї. Це двигун чатбота з відкритим вихідним кодом, який дозволяє визначати чати мовою, що базується на правилах [11].

Механізм ChatScript має потужний потік обробки природної мови та багату мову шаблонів. Він аналізує користувальницьку інформацію, теги, знаходить синоніми та концепції та знаходить відповідь, яка найкраще відповідає вхідній інформації. На додаток до функцій NLP, ChatScript також відстежує діалогові вікна, тому ви можете створювати довгі сценарії з різними темами. Але в ньому не буде нічого особливого, він не може запускати алгоритми машинного навчання, він не звертатиметься до зовнішніх баз знань або сторонніх інтерфейсів API, якщо ви самі не створите цю частину роботи.

Серйозна проблема з евристичним зіставленням шаблонів полягає в тому, що всі відповідні шаблони вимагають ручного програмування, що є непростим завданням, якщо чатбот повинен правильно обробляти сотні намірів. Ми можемо використовувати алгоритми машинного навчання для побудови моделі класифікації намірів. В такому випадку потрібен лише навчальний набір із сотень або тисяч прикладів, і модель вивчатиме шаблон наміру з цих даних [10].

Такий алгоритм може бути побудований за допомогою будь-якої популярної бібліотеки машинного навчання (наприклад, scikit-learn). Інший варіант – використовувати сторонні API-інтерфейси хмарної платформи, такі як wit.ai, api.ai та Microsoft LUIS. wit.ai, схоже, перша компанія, яка створила API машинного навчання для чатбота.

Алгоритми зіставлення зі зразком або класифікації машинного навчання можуть допомогти зрозуміти, що описує повідомлення, що

вводиться користувачем. Коли чатбот розуміє мету повідомлення, він генерує відповідь.

Чуйна архітектура вибору. Для однієї і тієї ж інформації чатбот може використовувати різні слова, щоб показати тональність. Наприклад, для метеорологічного робота чатбот може відповісти: «Буде дощ» або «80% часу буде дощ» або нагадати користувачам, що сьогодні ви повинні принести парасольку. Ніхто не знає, що різні користувачі мають різні почуття. Робот може аналізувати попередні чати та пов'язані показники (тривалість сеансу, ймовірність продажів, задоволеність користувачів), щоб вказати відповіді користувачів [13].

У чатбота може бути окремий модуль генерації та вибору відповіді. Конкретна архітектура показана на рисунку 1.7.

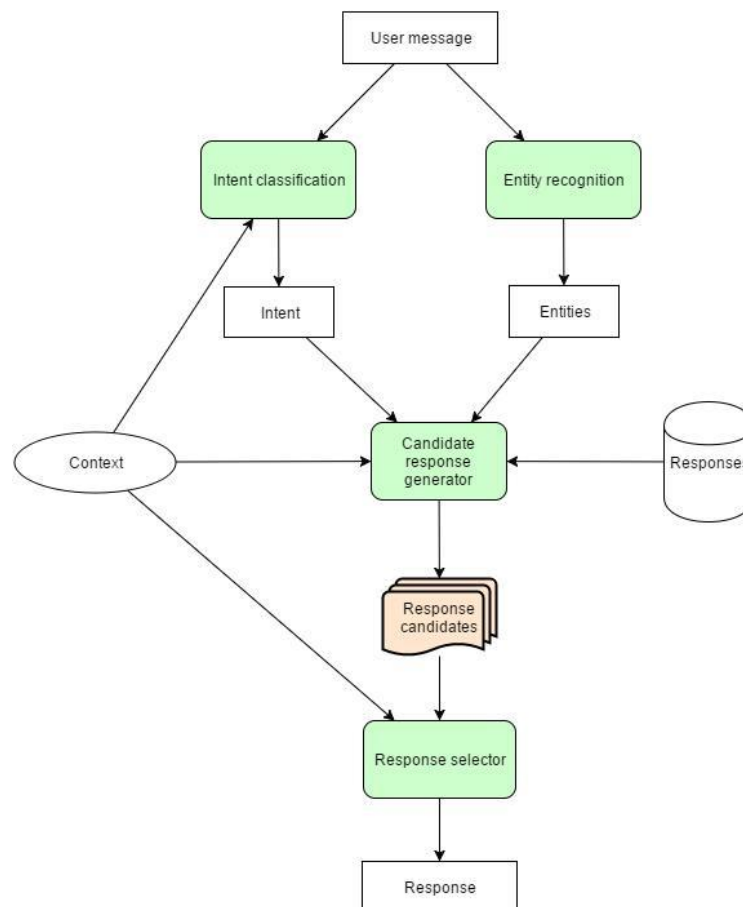


Рисунок 1.7 – Чуйна модель [13]

Обробка повідомлень зазвичай починається з розуміння того, що говорить користувач. Модуль класифікації намірів визначає справжнє намір повідомлення користувача. Зазвичай це вибір одного з кількох наперед визначених намірів. Хоча складніші роботи можуть розпізнавати кілька намірів з повідомлення, ми зазвичай проектуємо так, щоб розпізнавати один намір. Для класифікації намірів можемо використовувати контекстну інформацію, таку як намір попередніх повідомлень, особисту інформацію користувача та деякі переваги. Модуль розпізнавання сутностей витягує із повідомлення структуровану інформацію, наприклад, погодній роботі потрібно витягти місце та дату [13].

Генератор відповідей кандидата опрацьовує запити користувачів у всіх певних доменах. Він може використовувати різні алгоритми, викликати кілька зовнішніх API і навіть дозволяти людям допомагати із генерацією відповідей. Ці розрахунки є перелік можливих генераторів відповідей. Всі ці відповіді мають бути у певному полі, і логіка правильна. Це не може бути просто велика кількість випадкових відповідей. Генератор відповіді повинен використовувати контекст розмови та отримувати намір та сутність з останніх повідомлень користувача, інакше він не може підтримувати діалоги з кількома повідомленнями [14].

Селектор відповіді оцінює лише всі набори кандидатів відповіді та вибирає найкращу відповідь для користувача.

1.4 Висновки за розділом

Чатботи – це програми-співрозмовники, які імітують реальний усний чи письмовий діалог із користувачем. Автоматизацію спілкування з клієнтами останніми роками активно використовує бізнес. Великий сегмент ботів закриває завдання взаємодії всередині месенджера. Це як офіційні представництва та чати для спілкування з відомими людьми чи компаніями, так і додатковий функціонал для модерації різноманітних чатів та спільнот.

Розмовні чатбот – це програма на основі штучного інтелекту (ШІ), яка імітує людські розмови. Їх ще називають цифровими помічниками, котрі розуміють людські запити. Боти інтерпретують та обробляють запити користувачів та швидко дають відповідні відповіді.

Чатбот може генерувати відповіді з нуля на основі моделей машинного навчання або використовувати деякі евристики для вибору деяких відповідей із зарезервованої бази даних.

2 ТЕХНОЛОГІЇ І МЕТОДИ РОЗРОБКИ ЧАТБОТІВ

2.1 Технології розробки чатботів

Чатботи зазвичай створюються на будь-якій системі хмарних обчислень, яка підтримує мову програмування, що дозволяє розробникам створювати інтерактивний інтерфейс програмного забезпечення (API), наприклад, Java, Ruby, Rust, Node.js, Python, C++, Go і PHP [16]. Як зазначено в [15, 16], бекенд відповідає за збір вхідних даних, обробку неправильної обробки деяких існуючих API або алгоритмів і створення тексту відповіді.

У таблиці 2.1 показано порівняння різних платформ для створення чатботів.

Таблиця 2.1 - Порівняння різних платформ для створення чатботів

Платформа	Документація	Складність використання	Ціна	Мови програмування
MS Bot Framework	Добре	Складно	\$0.50 за 1000 повідомлень	.NET, JavaScript
Chartfuel	Добре	Легко	Безкоштовно	Python, PHP, JavaScript
Amazon Lex	Середньо	Складно	\$0.00075 за 1000 повідомлень	Python, PHP,
Telegram Bot	Добре	Легко	Безкоштовно	Python, PHP, Java

У цьому проєкті використовується API Telegram Bot, оскільки його документація щодо API є зрозумілою та простою, але з кількома корисними функціями. Його також можна використовувати для багатьох програм, таких як персоналізовані сповіщення та новини, приймати платежі користувачів Telegram та створювати соціальні сервіси.

На рисунку 2.1 показан алгоритм роботи чатбота Telegram.

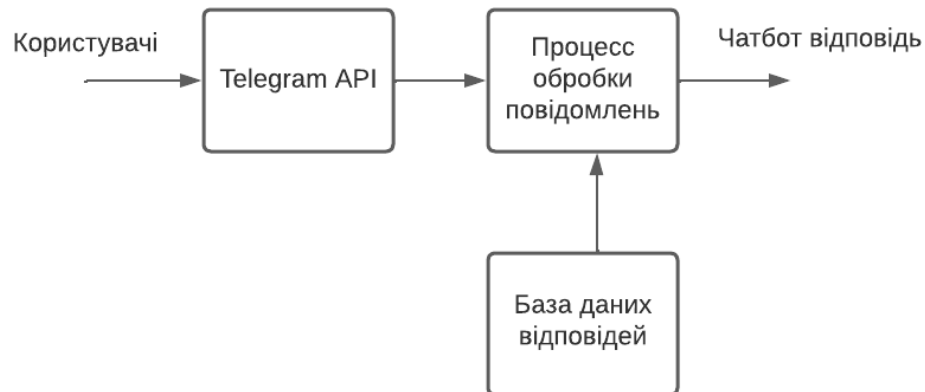


Рисунок 2.1 – Типовий алгоритм роботи чатботу Telegram

Чатботи зазвичай працюють, приймаючи введення користувача, яке потім надсилається в API чатбота. Потім чатбот обробляє дані шляхом пошуку попередньо визначеної моделі на основі правил або методів глибокого навчання, щоб забезпечити відповідну відповідь.

2.2 NLP-технології

Взагалі типовий цикл роботи будь-якого чатбота можна уявити ланцюжком наступних дій [10]:

- повідомлення від користувача;
- аналіз повідомлення, його обробка;
- виконання функцій згідно розробленого кейсу чатбота;
- надання відповіді прородною мовою;
- збереження повідомлення логів, статистики для наступних повідомлень;
- відправка відповіді в месенджер користувачу.

Найбільш складним етапом роботи є аналіз клієнтського запиту. Як ми вже відзначили раніше, чатботи, що самонавчаються, на базі Machine

Learning використовують для цього методи NLU і NLP. Наприклад, для текстових чатботів процес аналізу включає наступні етапи [11]:

- попередня обробка тексту – токенізація (розбиття на слова), виправлення друкарських помилок, лематизація та стемінг (визначення нормальної форми слів та частин мови), відкидання стоп-слів (артиклі, вигуки, спілки та ін.), розширення запиту за допомогою словників синонімів, доповнення інформацією про значимість окремих слів, розширення запиту деревом синтаксичного аналізу та результатами дозволу займенників, і навіть визначення іменованих сутностей.

- класифікація запиту на основі прикладів фраз та ML-алгоритмів або формальних правил (шаблонів); ранжування гіпотез класифікації відповідно до поточного контексту бесіди;

- вилучення параметрів запиту із фрази користувача.

Завдання інтерпретації людської мови та відповіді на неї наповнене безліччю проблем. Насправді, людям потрібні роки, щоб подолати ці проблеми та вивчити нову мову з нуля. Щоб подолати ці проблеми, програмісти інтегрували багато функцій у технологію NLP, щоб створити корисну технологію, яку можна використовувати для розуміння людської мови, процесу та отримання відповідної реакції.

Завдання NLP відповідають за розбивку людського тексту та аудіосигналів з голосових даних способами, які можна аналізувати та перетворювати в дані, які розуміє комп'ютер [15]. Деякі із завдань, які входять до прийому даних NLP, такі:

- розпізнавання мовлення: розпізнавання мовлення або перетворення мови в текст є неймовірно важливим процесом, залученим до аналізу мовлення. Позначення мовлення або граматичне тегування — це підпроцес розпізнавання мовлення, який дозволяє комп'ютеру розбивати мовлення та позначати його неявним контекстом, акцентом або іншими точками визначення мовлення;

- значення слів: у людській мові слово може мати кілька значень. Процес розшифровки значень слова — це семантичний аналіз, який вибирає значення даного слова, яке найкраще відповідає йому в даному контексті. Наприклад, цей процес допомагає визначити, чи є слово дієсловом чи займенником;

- розпізнавання названих сутностей або NEM: NEM визначає слова та фрази як корисні сутності, наприклад, «Руслан» — це ім'я людини, а «Америка» — назва країни [16];

- аналіз сентиментів: людська мова часто містить сентименти та відтінки. Вилучення цих відтінків і прихованих контекстів, таких як ставлення, сарказм, страх чи радість, є, мабуть, найскладнішим завданням, яке виконує процеси NLP.

Чатботи є відносно недавньою концепцією, і, незважаючи на наявність величезної кількості програм та інструментів NLP, ми в основному маємо лише дві різні категорії чатботів на основі технології NLP, яку вони використовують. Це два типи чатботів:

- скриптовані чатботи: чатботи зі сценаріями класифікуються як чатботи, які працюють із заздалегідь визначеними сценаріями, які створюються та зберігаються в їхній бібліотеці. Щоразу, коли користувач вводить запит або промовляє запит (у випадку чатботів, обладнаних модулями перетворення мови в текст), чатбот відповідає на цей запит відповідно до попередньо визначеного сценарію, який зберігається в його бібліотеці. Одна з мінусів такий чатбот полягає в тому, що користувачу необхідно надати свій запит дуже структуровано за допомогою команд, розділених комами, або інших форм регулярного виразу, що полегшує роботу бота для виконання аналізу рядків і розуміння запиту. Це ускладнює інтеграцію такого типу чатботів із модулями перетворення мови в текст за допомогою NLP. Тому ці чатботи навряд чи можна перетворити на розумних віртуальних помічників [15];

- чатботи зі штучним інтелектом: як впливає з назви, чатботи зі штучним інтелектом створені, щоб імітувати людські риси та реакції. NLP або Обробка природної мови несе величезну відповідальність за те, щоб такі чатботи могли розуміти діалекти та відтінки людської розмови. NLP у поєднанні зі штучним інтелектом створює справді розумного чатбота, який може відповідати на тонкі запитання та вчитися на кожній взаємодії, щоб наступного разу створювати кращі відповіді. Чатботи зі штучним інтелектом були розроблені, щоб допомагати користувачам на різних платформах, таких як автоматична підтримка чату або віртуальні помічники, які допомагають з вибором пісні або ресторану [11].

NLP містить багато цікавих бібліотек, найбільшою з яких є NLTK (Natural Language Toolkit), ця бібліотека досить універсальна, але також досить складна у використанні. У більшості випадків вона досить повільна і не відповідає вимогам виробництва, що швидко розвивається. Інші відомі бібліотеки: TextBlob, CoreNLP, Polyglot, SpaCy, Gensim.

2.4 Проєктування чатботу

З рисунку 2.2 користувач ініціалізує розмову або просто задає запит за допомогою тексту. Запит буде надходити до чатбота через графічний інтерфейс користувача. Іноді користувачі часто роблять помилки ненавмисно. Запит або дані можуть бути чи не законними. Це означає, що в запиті може бути присутність шуму, помилки, спеціальних символів або орфографічні помилки. Це дуже вірно в людській природі. Запит надсилається природною мовою обробка для уточнення помилок або шуму. З блоку обробки природної мови, дані вводяться в блок обробки алгоритму машинного навчання. Цей пристрій шукатиме відповідь у бази знань за допомогою зіставлення шаблонів. Відповідь буде обрано. Вибрана відповідь буде надіслана користувачеві текстом. Припустимо, що відповідь не

знайдено за допомогою відповідності шаблону, відобразиться відповідь за замовчуванням, тобто «Я не розумію».

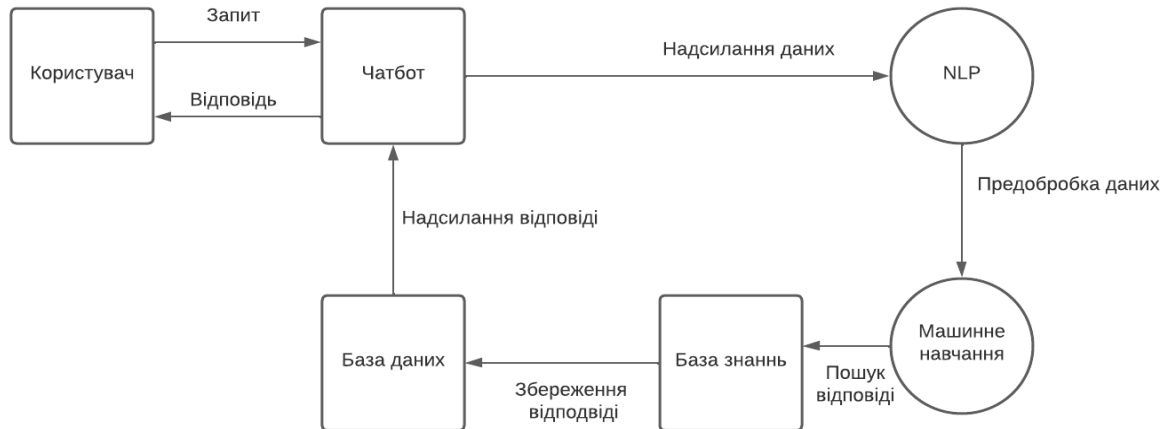


Рисунок 2.2 – Ініціалізація розмови з чатботом

Діаграма варіантів використання описує взаємодію користувача з системою. Корпуси овальної форми `shape` описують очікувану поведінку користувача та системи. Наступний рисунок 2.3 описує схему використання чатбота. На діаграмі є два актори: користувач і чатбот. Взаємодії, такі як звичайний чат або розмова, щоб задати запитання та щоб дати відповідь на цей запит. Припустимо, відповідь на запит не знайдена в базі даних чатбот дасть відповідь за замовчуванням, тобто «Я не розумію».

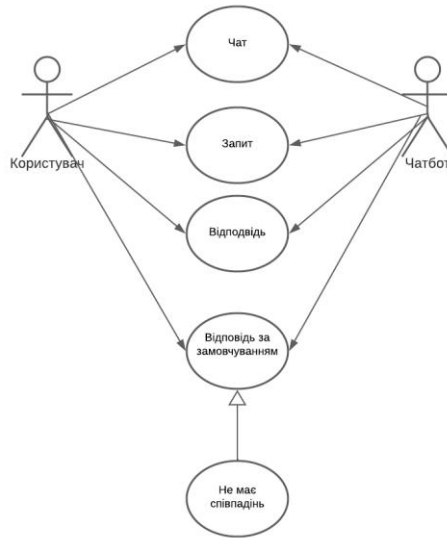


Рисунок 2.3 – Діаграма використання чатботу

Діаграма діяльності — це блок-схема діяльності. Вона представляє робочий процес між різними діями. Її також називають об’єктно-орієнтованою блок-схемою. Це схоже на блок-схему, яка візуалізує перехід від однієї діяльності до іншої (рис. 2.4).

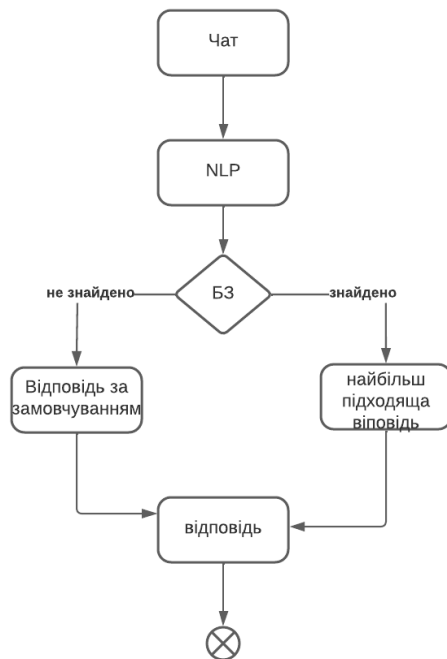


Рисунок 2.4 – Діаграма станів чатботу

2.4 Висновки за розділом

У цьому проєкті використовується API Telegram Bot, оскільки його документація щодо API є зрозумілою та простою, але з кількома корисними функціями. Чатботи зазвичай працюють, приймаючи введення користувача, яке потім надсилається в API чатбота. Потім чатбот обробляє дані шляхом пошуку попередньо визначеної моделі на основі правил або методів глибокого навчання, щоб забезпечити відповідну відповідь.

Чатботи є відносно недавньою концепцією, і, незважаючи на наявність величезної кількості програм та інструментів NLP, ми в основному маємо лише дві різні категорії чатботів на основі технології NLP, яку вони використовують. Це два типи чатботів: скриптові та із штучним інтелектом.

Ініціалізація розмови з чатботом виглядає наступним чином. Запит буде надходити до чатбота через графічний інтерфейс користувача. Іноді користувачі часто роблять помилки ненавмисно. Запит або дані можуть бути чи не законними. Це означає, що в запиті може бути присутність шуму, помилки, спеціальних символів або орфографічні помилки. Це дуже вірно в людській природі. Запит надсилається природною мовою обробка для уточнення помилок або шуму. З блоку обробки природної мови, дані вводяться в блок обробки алгоритму машинного навчання. Цей пристрій шукатиме відповідь у бази знань за допомогою зіставлення шаблонів. Відповідь буде обрано. Вибрана відповідь буде надіслана користувачеві текстом. Припустимо, що відповідь не знайдено за допомогою відповідності шаблону, відобразиться відповідь за замовчуванням, тобто «Я не розумію».

В даному рзділі спроектовані UML-діаграми використання та станів для відображення роботи чатбота.

3 ОПИС ПРОГРАМИ

3.1 Попередня обробка тексту

Попередня обробка текстовий повідомлень починається з видалення символів пунктуації та багатої кількості прогалин:

```
#розмітка слів
def tag(this, words):
    return this.post.tag(words)
def canonize_text(this, s):
    #видалення більше двох прогалин
    s = re.sub("(\s{2,})", ' ', s.strip())
    return s
def removeTermin(this, s):
    #видалення пунктуації
    return s[:-1].strip() if s[-1] in '?!.'
else s
```

Невеликий варіант стеммінга - лемматизація. Основна відмінність між ними полягає в тому, що коріння часто може створювати неіснуючі слова, тоді як лєми є справжніми словами:

```
class Lemmat(object):
    def __init__(this):
        this.lemma = Mystem()
    @staticmethod
    def isGoodToken(t):
        return len(t.strip()) > 0 and not t[0] in
!?'- '
    def token(this, phrase):
        lemma = [lemma for lemma in
this.lemma.lemmat(phrase) if Lemmat.isGoodToken(lemma)]
```

```
return lemma
```

Надалі всі оброблені слова збираються у мішок слів:

```
def splitCl(this, s):
    return list(this.Split.split(s))
def tokenizes(this, s):
    return this.tokenizer.tokenizes(s)
def parseSyntax(this, str):
    processed = this.pipLine.process(str,
this.pipError)
    if this.pipError.occurred():
        logging.error("An error occurred when
running run_udpipe: %s", this.pipError.message)
        return None
    parsed_data =
pyconll.load_from_string(processed)[0]
    return parsed_data
```

Розробка детектору синонімів є дуже важливою частиною обробки природної мови:

```
from botModel import ModelApplicator
class SynonymyDetector(ModelApplicator):
    def __init__(self):
        super(SynonymyDetector, self).__init__()
    def get_most_similar(self, probe_phrase,
phrases, text_utils, nb_results=1):
        raise NotImplementedError()
    def calc_synonymy2(self, phrase1, phrase2,
text_utils):
        raise NotImplementedError()
```

```
def get_threshold(self):
    return 0.7
```

3.2 Розробка чатботу

Для реалізації робота будемо використовувати Python 3.7. Це високорівнева мова програмування з динамічною типізацією, що підтримує об'єктно-орієнтований, функціональний та імперативний стилі програмування, стандартна бібліотека включає великий обсяг корисних функцій. Ця мова відмінно підходить для написання роботів, він містить відповідні для цього бібліотеки: "telebot" і "requests" [12].

Оскільки неможливо весь час тримати комп'ютер увімкненим, знадобиться хмарний сервіс “Heroku” [13], на який буде завантажено бот. Heroku - хмарна PaaS-платформа, що підтримує ряд мов програмування. Це хмарне масштабоване серверне рішення, яке дозволяє легко керувати розгортанням ваших Rails (або інших) програм за умови підписання низки угод (наприклад, Postgres як база даних, без запису в файлову систему).

Для створення чатботів у Telegram можете використовувати бібліотеки python-telegram-bot або pyTelegramBotAPI.

Принципово ці бібліотеки нічим одна від одної не відрізняються. Відмінність пов'язана із синтаксисом.

Якщо мова йде про чатбот для сотень людей, то краще використовувати бібліотеку aiogram. На відміну від двох попередніх бібліотек aiogram підтримує асинхронність. Це дозволить обробляти повідомлення кількох людей одночасно.

Базовий клас для процесу розмови між чатботом та користувачем:

```
class BaseAnsweringMachine(object):
    def __init__(self):
        pass
    def get_session_factory(self):
```

```

        raise NotImplementedError()
    def start_conversation(self):
        pass
    def push_phrase(self, bot, interlocutor,
phrase):
        pass
    def pop_phrase(self, bot, interlocutor):
        return u''
    def get_session(self, interlocutor):
        return
self.get_session_factory()[interlocutor]

```

Налаштування профілю бота:

```

class BotProfile(object):
    def __init__(this, bot_id=''):
        this.botId = botId
        this.prof = None
        this.premisesPath = None
        this.faqPath = None
        this.constants = dict()
    def getId(this):
        return this.botId
    def replace(this, pathStr, dataDir,
modelsDir):
        return pathStr.replace('$DATA',
dataDir).replace('$MODELS', modelsDir)
    def loading(this, profilePath, dataDir,
modelsDir):
        with open(profilePath, 'r') as f:
            this.profile = json.load(f)

```

```

        this.premisesPath =
this.replace(this.profile['premises'],      dataDir,
modelsDir) if 'premises' in this.profile else None
        this.faqPath =
this._replace(this.profile['faq'],          dataDir,
modelsDir) if 'faq' in this.profile else None
        this.constants = this.profile['constants']

```

Історія діалогу зберігається в окремі файли. В подальшому її можна використати для навчання моделі ШІ.

Основна проблема з текстовими даними полягає в тому, що всі вони мають текстовий формат (рядки). Однак алгоритмам машинного навчання потрібен якийсь числовий вектор ознак, щоб виконати завдання. Основна попередня обробка тексту включає [5]:

- перетворення всього тексту у верхній або нижній регістр, щоб алгоритм не розглядав одні й ті самі слова в різних регістрах як різні;
- токенізація: Токенізація — це лише термін, який використовується для опису процесу перетворення звичайних текстових рядків у список маркерів, тобто слів, які ми насправді хочемо. Токенізатор речень можна використовувати для пошуку списку речень, а токенізатор слова — для пошуку списку слів у рядках;
- видалення шуму, тобто всього, що не містить стандартних цифр або літер;
- видалення стоп-слів. Іноді деякі надзвичайно поширені слова, які, здавалося б, мають невелику цінність у виборі документів, які відповідають потребам користувача, повністю виключаються зі словника. Ці слова називаються стоп-словами;
- створення коренів: це процес скорочення відмінених (або іноді похідних) слів до їх основи, основи або кореня — як правило, письмової форми слова. Приклад, якби ми поставили основу для наступних слів:

«Стовбури», «Стовбурові», «Стовбурові», «і стемтизація», результатом буде одне слово «стебло»;

- лемматизація.

Після початкової фази попередньої обробки нам потрібно перетворити текст у значущий вектор (або масив) чисел. Мішок слів — це представлення тексту, що описує присутність слів у документі. Воно включає дві речі:

- словник відомих слів;
- міра наявності відомих слів.

Розробка датасету проводилась вручну. Інтенти — наміри користувачів. Інтент включає приклади питань, які задають користувачі. Наприклад, інтент Спортивні заходи містить усі питання (власне кажучи, це можуть бути і твердження), пов'язані зі спортивними заходами. Також інтент включає відповіді чатбота. Інтент може містити одну або кілька відповідей. Якщо відповідей кілька, то відповідь робота вибирається випадковим чином.

Зчитування фактів проходить в два етапи. Окремо записані факти у файл та у другий файл їх ймовірність появи.

Приклад опису фактів чатботу у Додатку Б.

Начання моделі проводилось за допомогою бібліотек tensorflow та Kernes:

```
class NN_Model(PremisesModel):
    def __init__(this):
        super(NN_Model, this).__init__()
    def load(this, modelFolder):
        with open(os.path.join(modelFolder,
'nn_model.config'), 'r') as f:
            modelConfig = json.load(f)
            this.mexLen = modelConfig['mexLen']
            this.maxNB = modelConfig['maxNB']
            this.tokenIndex =
modelConfig['tokenIndex']
```

```

        this.filePath =
this.gettModel(modelFolder, modelConfig['filePath'])
        this.weightsFile =
this.gettModel(modelFolder, modelConfig['weightsPath'])

        this.bpeModel =
spm.SentencePieceProcesor()
        rc =
this.bpeModel.Load(this.gettModel(modelFolder,
modelConfig['bpeModelName']+'.model'))

        with open(this.filePath, 'r') as f:
            m = modelJson(f.read())
            m.loadW(this.weightsFile)
            this.model = m
            this.XnProbe = []
            for _ in range(this.maxNB+1):
                x = np.zeros((1, this.mexLen),
dtype=np.int32)
                this.XnProbe.append(x)
            this.inputs = dict()
            for premise in range(this.maxNB):
this.inputs['premise{}'.format(premise)] =
this.XnProbe[premise]
                this.inputs['question'] =
this.XnProbe[this.maxNB]

```

Повний код навчання моделі наведений в Додатку В.

3.3 Висновки за розділом

Попередня обробка текстовий повідомлень починається з видалення символів пунктуації та багатої кількості прогалин. Невеликий варіант стеммінга - лемматизація. Основна відмінність між ними полягає в тому, що коріння часто може створювати неіснуючі слова, тоді як леми є справжніми словами.

Для реалізації робота будемо використовувати Python 3.7. Це високорівнева мова програмування з динамічною типізацією, що підтримує об'єктно-орієнтований, функціональний та імперативний стилі програмування, стандартна бібліотека включає великий обсяг корисних функцій. Оскільки неможливо весь час тримати комп'ютер увімкненим, знадобиться хмарний сервіс “Heroku”, на який буде завантажено бот. Для створення чатботів у Telegram можете використовувати бібліотеки `python-telegram-bot`. Начання моделі ШІ проводилось за допомогою бібліотек `tensorFlow` та `Kernes`.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Реєстрація чатботу

Насамперед необхідно зареєструвати бота та отримати його токен (ключ для авторизації бота та взаємодії з ним), написавши боту під ім'ям @BotFather.. (рис. 4.1, рис. 4.2).

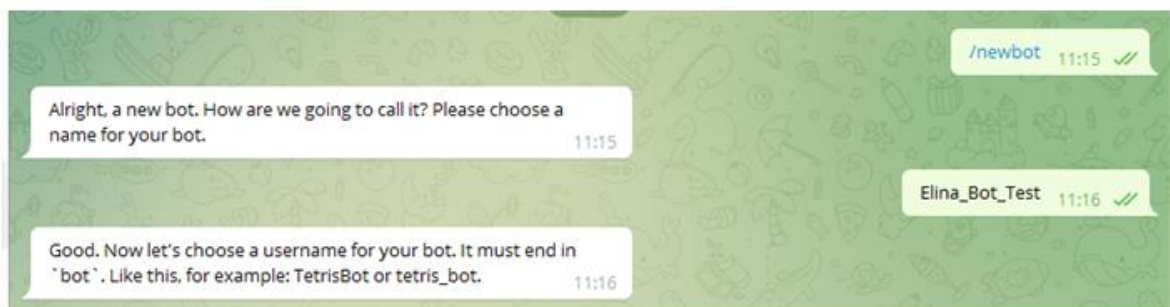


Рисунок 4.1 – Реєстрація імені чатботу

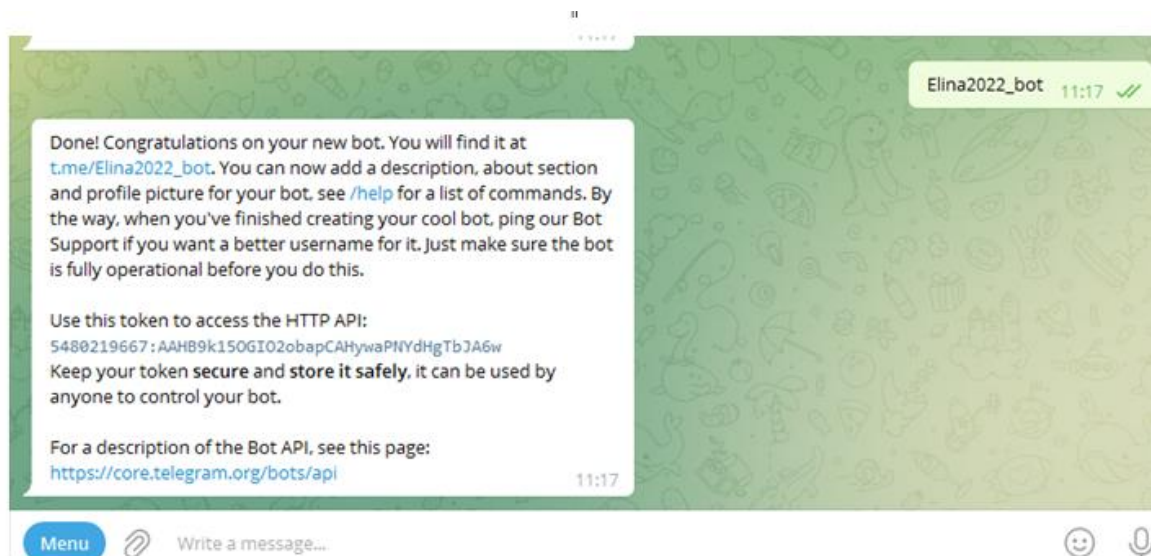


Рисунок 4.2 – Отримання токена чатботу

У відповідь від BotFather ми отримали токен – ідентифікатор чатбота, який складається з цифр та символів.

4.2 Тестування чатботу

Щоб авторизуватися в роботі, для початку його потрібно буде знайти в пошуковій системі. Після цього написати «Привіт» запустити робота (рис. 4.3).

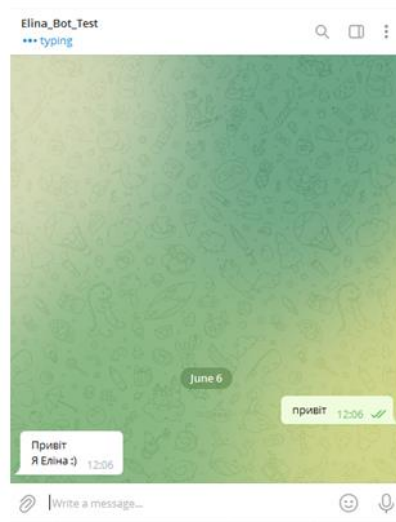


Рисунок 4.3 – Запуск розмови чатботу

Подальше спілкування з чатботом у довільній формі. Перше правило чатбота, він запитує ім'я людини. Для того,щоб наділі правильно звертатись (рис. 4.4).

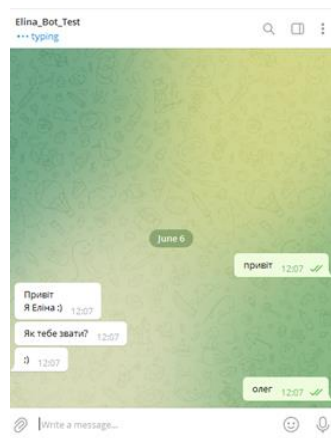


Рисунок 4.4 – Знайомство зі співрозмовником

Далі протестуємо відповіді чатбота на одну ситуацію з різним розвитком подій (рис. 4.5, рис. 4.6, рис. 4.7).



Рисунок 4.5 – Моделювання ситуації розмови



Рисунок 4.6 – Моделювання ситуації розмови. Відповідь позитивна



Рисунок 4.7 – Моделювання ситуації розмови. Відповідь негативна

Надалі, для вдосконалення цього бота можна зробити обробку голосових повідомлень. У спілкуванні ми використовуємо як текстові, а й голосові повідомлення, тому можливість відповідати голосові повідомлення буде необхідною функцією для чатбота.

4.4 Висновки за розділом

Насамперед необхідно зареєструвати бота та отримати його токен (ключ для авторизації бота та взаємодії з ним), написавши боту під ім'ям @BotFather. У відповідь від BotFather ми отримали токен – ідентифікатор чатбота, який складається з цифр та символів.

Щоб авторизуватися в роботі, для початку його потрібно буде знайти в пошуковій системі. Після цього написати «Привіт» запустити робота. Подальше спілкування з чатботом у довільній формі. Перше правило чатбота, він запитує ім'я людини. Для того, щоб наділі правильно звертатись. Протестовано відповіді чатбота на одну ситуацію з різним розвитком подій.

ВИСНОВКИ

В кваліфікаційній роботі був розроблен чатбот обробки природної мови. Актуальність розробки даного бота обумовлена автоматизацією спілкування з клієнтами. Великий сегмент ботів закриває завдання взаємодії всередині месенджера. Це як офіційні представництва та чати для спілкування з відомими людьми чи компаніями, так і додатковий функціонал для модерації різноманітних чатів та спільнот.

У цьому проєкті використовується API Telegram Bot, оскільки його документація щодо API є зрозумілою та простою, але з кількома корисними функціями. Ініціалізація розмови з чатботом виглядає наступним чином. Запит надходить до чатбота через графічний інтерфейс користувача. З блоку обробки природної мови, дані вводяться в блок обробки алгоритму машинного навчання. Цей пристрій шукатиме відповідь у бази знань за допомогою зіставлення шаблонів. Відповідь буде обрано. Вибрана відповідь буде надіслана користувачеві текстом. Припустимо, що відповідь не знайдено за допомогою відповідності шаблону, відобразиться відповідь за замовчуванням, тобто «Я не розумію».

Для реалізації робота будемо використовувати Python 3.7. Це високорівнева мова програмування з динамічною типізацією, що підтримує об'єктно-орієнтований, функціональний та імперативний стилі програмування, стандартна бібліотека включає великий обсяг корисних функцій. Оскільки неможливо весь час тримати комп'ютер увімкненим, знадобиться хмарний сервіс “Heroku” , на який буде завантажено бот. Для створення чатботів у Telegram використана бібліотека python-telegram-bot. Начання моделі III проводилось за допомогою бібліотек tensorFow та Kernes.

Всі поставлені задачі було виконано та мета проєкту досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Adam M. AI-based chatbots in customer service and their effects on user compliance / M. Adam // Electronic Markets - vol. 9 - no. 2. – 2020. - p. 204.
2. Використання технології чат-ботів в умовах цифрової трансформації бізнесу [Електрон. ресурс]. - Режим доступу: <http://confmanagement.kpi.ua/proc/article/view/230945>
3. Чат-боти у навчанні: 7 ідей для їх використання [Електрон. ресурс]. - Режим доступу: <https://learnlifelong.net/chat-boty-u-navchanni-7-idej-dlya-yih-vykory/>
4. Переваги та недоліки використання чат-ботів для бізнесу [Електрон. ресурс]. - Режим доступу: <https://internetdevels.ua/blog/pros-and-cons-of-using-chatbots-for-business>
5. Sperli G. A Cultural heritage framework using a Deep Learning based Chatbot for supporting tourist journey / G. Sperli // Expert Systems with Applications. – N.4. – 2021. - p. 115277.
6. Прокопенко Т., Обойщик О. Особливості використання чатботів для бізнесу у сучасних месенджер чатах / Т. Прокопенко, О. Обойщик // Вісник Черкаського державного технологічного університету. Серія: Технічні науки. – 2019. С. 11-16. 10.24025/2306-4412.1.2019.165418.
7. Щербаков М.В. Целевой чат-бот на основе машинного обучения / М.В. Щербаков. – 2020 - 10.26102/2310-6018/2020.29.2.041.
8. Лавренчук С., Товстенюк Б. Інформаційний чат-бот для сервісу обміну повідомленнями Telegram в навчальній сфері / С. Лавренчук, Б. Товстенюк // Computer-Integrated technologies^ Education, Science, Production.- 2020. – С. 180-185.
9. Сатыбалдина Д., Глазырина Н., Рсалиев Ж. Разработка автоматизированной системы управления работой образовательного центра // Bulletin of Toraighyrov University. Energetics series. – 2020. – С. 303-314.

10. Dhyani M., Kumar. R. An intelligent Chatbot using deep learning with Bidirectional RNN and attention model / . M. Dhyani, R. Kumar // Materials today: proceedings. - vol. 34. – 2021. - pp. 817-824.
11. Що таке чат-бот: секрети використання та основні переваги для бізнесу [Електрон. ресурс]. - Режим доступу: <https://helpcrunch.com/blog/uk/shcho-take-chat-bot/>
12. Fahad S.A. Inflectional review of deep learning on natural language processing / S. A. Fahad and A. E. Yahya // in 2018 international conference on smart computing and electronic enterprise (ICSCEE). - 2018: IEEE. - pp. 1-4.
13. Raj S. Building Chatbots with Python: Using natural language processing and machine learning / S. Raj. - Apress, 2019. – 34 p.
14. Mordzyk N. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API / N. Modrzyk. - Apress, 2019. – 24 p.
15. A simple, but extensible Python implementation for the Telegram Bot API – Python Telegram bot api [Electronic resource]. – Access mode: <https://github.com/eternnoir/pyTelegramBotAPI>
16. Heroku Dev Center. Page provides all published technical documentation for the Heroku platform [Electronic resource]. – Access mode: <https://devcenter.heroku.com/categories/reference>
17. Copyright, Python Software Foundation. Python 3.7.4 documentation [Electronic resource]. – Access mode: <https://docs.python.org/3/>
18. Як користуватися чат-ботами для навчання та навчання [Електрон. ресурс]. - Режим доступу: <https://www.chatcompose.com/uk/learning.html>
19. Вплив чат-ботів на веб-розробку [Електрон. ресурс]. - Режим доступу: <https://www.zfort.com.ua/blog/vpliv-chat-botiv-na-veb-rozrobku>

ДОДАТОК А
Текст програми

Клас зчитування фактів

```

class FactRead(torage):
    def __init__(this, txtUtils, path, const):
        super(FactRead,
this).__init__(txtUtils)
        this.txtUtils = txtUtils
        this.profileFact = None
        this.const = const
        this.newFact =
collections.defaultdict(list)

    def loadProfile(this):
        if this.profileFact is None:
            this.profileFact = []
            if this.path is not None:
                with io.open(this.path, 'r',
encoding='utf=8') as rdr:
                    currSetction = None
                    for lines in rdr:
                        lines = lines.strip()
                        if 'profSection:'
in lines:
                            elif
'import' in lines:
                                fn =
re.search('import "(.+)"', lines).group(1).strip()
                                addPath =
os.path.join(os.path.dirname(this.path), fn)

this.logger.debug('Loading facts from file "%s"...',
addPath)
                                with
io.open(addPath, 'rt', encoding='utf-8') as rdr2:
                                    for
lines in rdr2:
                                        lines = lines.strip()
if
lines and not lines.startswith('#'):
lines1 = random.choice(lines.split('|')).strip()

```

```

canonLina = this.txtUtils.canonize_text(lines1)

this.profileFact.append((canonLina,          currSetction,
addPath))

                                else:

                                continue
                                else:

assert(currSetction)

                                lines1          =
random.choice(lines.split('|')).strip()
                                canonLina        =
this.txtUtils.canonize_text(lines1)
                                canonLina        =
replace_constant(canonLina, this.const, this.txtUtils)

this.profileFact.append((canonLina,          currSetction,
this.path))

    def resetAddedFacts(this):
        this.newFact =
collections.defaultDict(list)
    def resetAllFacts(this):
        this.resetAddedFacts()
        this.profileFact = None

    def getAddedFacts(this, interlocutor):
        return this.newFact[interlocutor]

    def findTaggedFact(this,          interlocutor,
factTag):
        for fact,          section,          tag          in
this.newFact[interlocutor]:
            if tag == factTag:
                return fact

        return None

```

ДОДАТОК Б
Приклад фактів

У чатботі використовується машинне навчання
у чатботів немає батьків
у роботів немає батьків
у роботів не буває дітей
у робота не може бути мами
у робота не може бути тата
у людей завжди є батьки
у людини завжди є мама
у людини завжди є тато
У баранів є роги
У оленів є роги
У лосів є роги
У ведмедів немає рогів
Коти не мають рогів
У собак немає рогів
Комари п'ють кров
Сліпі п'ють кров
П'явки п'ють кров
хижі тварини їдять м'ясо інших тварин
леви це хижі тварини
вовки це хижі тварини
лисиці це хижі тварини
куниці це хижі тварини
Горобці цвірінькають
Чірікання видають горобці
Кішки нявкають
кішки не гавкають
М'якання видають кішки
Собаки гавкають
собаки гавкають
собаки не нявкають
собаки не вміють шипіти
собаки кусаються
собаки дуже віддані своєму господарю
гавкіт видають собаки
з пістолета можна стріляти у ворога
Кішки можуть шипіти
Шипіння видають кішки
Змії можуть шипіти
Шипіння видають змії
Кури квохчуть
Квохтання видають кури
півні кукурікають
кукурікання видають півні
Свині рохкають

корови їдять сіно
горобці не їдять риби
корови не їдять риби
коні не їдять риби
кози не їдять риби
коні іржуть
кішки нявкають
коти нявкають
корови мукають
вовки виють
вовки їдять м'ясо
вовки ловлять зайців
лисиці ловлять зайців
лисиці ловлять мишей
вовки не лазять по деревах
лисиці руді
вовки сірі
у кішок різне забарвлення
у собак різне забарвлення
вовки не літають
лисиці не літають
метелики літають
метелики літають
комарі літають
комарі не вміють бігати
метелики не вміють бігати
метелики не вміють бігати
гусениці не вміють бігати
гусениці харчуються листям
кішки - тварини
собаки - тварини
коні - тварини
корови - тварини
лисиці - тварини
горобці - птахи
гусениці - комахи
метелики - комахи
метелики - комахи
комарі - комахи
жуки - комахи
у павуків вісім лап
у комах шість лап
у тварин чотири лапи
у собаки два очі
у кішки чотири лапи

ДОДАТОК В
Навчання моделі

```

class NN_Model(PremisesModel):

    def __init__(this):
        super(NN_Model, this).__init__()

    def load(this, modelFolder):

        with open(os.path.join(modelFolder,
'nn_model.config'), 'r') as f:
            modelConfig = json.load(f)

            this.mexLen = modelConfig['mexLen']
            this.maxNB = modelConfig['maxNB']
            this.tokenIndex =
modelConfig['tokenIndex']
            this.filePath =
this.gettModel(modelFolder, modelConfig['filePath'])
            this.weightsFile =
this.gettModel(modelFolder, modelConfig['weightsPath'])

            this.bpeModel =
spm.SentencePieceProcesor()
            rc =
this.bpeModel.Load(this.gettModel(modelFolder,
modelConfig['bpeModelName']+'.model'))

            with open(this.filePath, 'r') as f:
                m = modelJson(f.read())

            m.loadW(this.weightsFile)
            this.model = m

            this.XnProbe = []
            for _ in range(this.maxNB+1):
                x = np.zeros((1, this.mexLen,),
dtype=np.int32)
                this.XnProbe.append(x)

            this.inputs = dict()
            for premise in range(this.maxNB):

```

```

this.inputs['premise{}'.format(premise)]           =
this.XnProbe[premise]
            this.inputs['question']               =
this.XnProbe[this.maxNB]

    def      isYnough(this,      strList,      queSTR,
textUtils):
    assert(len(strList) <= this.maxNB)
    assert(len(queSTR) > 0)

        for      premise,      premise      in
enumerate(strList):
            tx                                           =
this.bpeModel.EncodeAsPieces(premise)
            for      tokeni,      token      in
enumerate(tx[:this.mexLen]):
                this.XnProbe[premise][0, tokeni] =
this.tokenIndex.get(token, 0)

            tx = this.bpeModel.EncodeAsPieces(queSTR)
            for      tokeni,      token      in
enumerate(tx[:this.mexLen]):
                this.XnProbe[this.maxNB][0, tokeni] =
this.tokenIndex.get(token, 0)

    y = this.model.predict(x=this.inputs)[0]

    pEnough = y[0]
    return pEnough

```

ДОДАТОК Г
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

РОЗРОБКА ЧАТБОТА ОБРОБКИ ПРИРОДНОЇ МОВИ.

Виконала студентка 4 курсу, групи КНТ-218 Гадун Е.В.

Керівник к.т.н. доцент

Скрупський С.Ю.

Рисунок Г.1 – Слайд 1

Мета та актуальність теми

Мета проекту – це розробка чатботу обробки природної мови. Актуальність обраної теми складається з того, що віртуальне спілкування є дуже значимі для людини. В кожного користувача інтернетом є більше двох видів додатків для комунікації. Таким чином, сучасні клієнтоорієнтовані інформаційні системи все частіше мають у своєму складі чатботи в різних месенджерах для мовного спілкування та обміну інформацією.

2/11

Рисунок Г.2 – Слайд 2

Таблиця 1 – Аналіз сучасних чатботів

Особливості чатботів	- найчастіше мінімальна можливість взаємодії з файловою системою та операційною системою користувача; - обмеженість інтерфейсу вибраним чат-додатком; - залежність від працездатності зовнішнього продукту (це дуже актуально останнім часом); - максимально утруднене автоматизоване тестування.
Переваги	- користувач знаходиться в знайомому та зрозумілому для нього інтерфейсі; - можливість масштабування.
Недоліки	- це конструювання зручного та функціонального інтерфейсу вже на базі існуючого; - різноманітне отримання складного сценарію поведінки користувача в багатьох месенджерах.

3/11

Рисунок Г.3 – Слайд 3

Розмовні чатботи

РЧ аналізує запит користувача, щоб дати найбільш релевантну відповідь на цей запит – це є його ключовим завданням. Після завершення аналізу користувач отримує відповідь на запит. РЧ працюють із використанням трьох методів класифікації:

- зіставлення з шаблоном. Боти використовують збіги шаблонів для групування тексту, щоб вибрати відповідну відповідь. Мова розмітки штучного інтелекту (AIML) – це стандартна структурована модель цих шаблонів. Бот може отримати правильну відповідь, порівнявши запит із шаблоном. Таке зіставлення вимагає багато попередньо згенерованих шаблонів.

- розуміння природної мови (NLU). NLU (Natural language understanding, розуміння природної мови) - це здатність РЧ розуміти людину. Це процес перетворення тексту на структуровані дані для розуміння машиною. NLU слідує трьом конкретним концепціям: сутності, контекст та очікування.

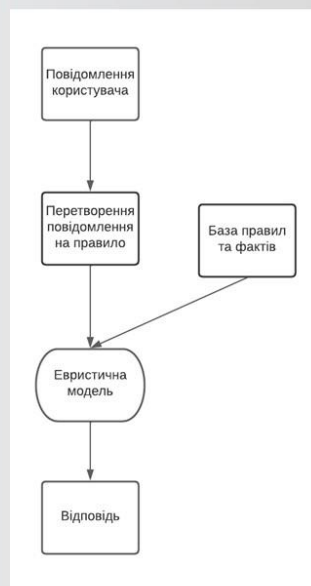
- обробка природної мови (NLP). Боти для обробки природної мови (NLP) призначені для перетворення тексту або мови користувача в структуровані дані. Дані надалі використовуються для вибору відповідної відповіді.

4/11

Рисунок Г.4 – Слайд 4

Модель чатботу

Моделі відповідей чатботу: генеративна модель, модель на основі пошуку, евристична модель та чуйна модель вибору.



5/11

Рисунок Г.5 – Слайд 5

Структура NLP - боту



6/11

Рисунок Г.6 – Слайд 6

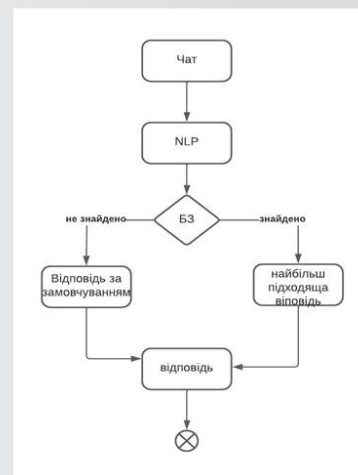
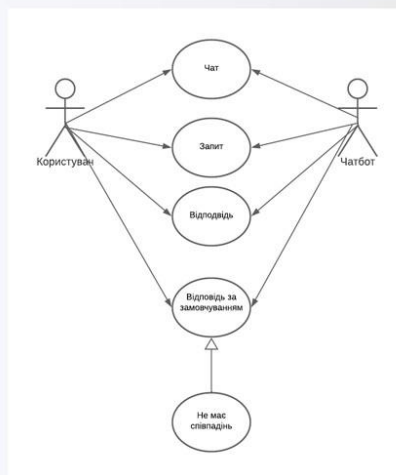
Таблиця 2- Порівняння різних платформ для створення чатботів

Платформа	Документація	Складність використання	Ціна	Мови програмування
MS Bot Framework	Добре	Складно	\$0.50 за 1000 повідомлень	.NET, JavaScript
Chartfuel	Добре	Легко	Безкоштовно	Python, PHP, JavaScript
Amazon Lex	Середньо	Складно	\$0.00075 за 1000 повідомлень	Python, PHP,
Telegram Bot	Добре	Легко	Безкоштовно	Python, PHP, Java

7/11

Рисунок Г.7 – Слайд 7

UML-діаграми чатботу



8/11

Рисунок Г.8 – Слайд 8

Тестування чатботу

Рисунок Г.10 – Слайд 10



Висновки

В ході роботи були вирішені наступні завдання:

- проведено аналіз сучасних чатботів та платформ для їх реалізації;
- обрано технології для розробки чатбота обробки природної мови;
- розроблен розмовний чатбот;
- проведено тестування чатботу.

Для реалізації робота будемо використовувати Python 3.7.

Для створення чатботів у Telegram використана бібліотека `python-telegram-bot`. Начання моделі ШІ проводилось за допомогою бібліотек `tensorflow` та `Keras`.



Рисунок Г.11 – Слайд 11