МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 23.00.00.000 ПЗ

Група ІПм-23-1

Глушко Михайло

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Глушко Михайло Сергійович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми інтеграції чат-ботів для автоматизації внутрішніх процесів в компаніях

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Глушко М.С.
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Козак Олексій Федорович, к.т.н., старший викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри
доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль
доц. Вовк Р.Б.
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Глушку Михайлу Сергійовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “Моделі, методи та алгоритми інтеграції чат-ботів для автоматизації внутрішніх процесів в компаніях**”**
керівник проекту (роботи) Козак Олексій Федорович, к.т.н., старший викладач
затверджені наказом закладу вищої освіти від **“** 09 ” листопада 2024 р. № 561/7
2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.
3. Вихідні дані до проекту (роботи) Архітектура, формальний опис та алгоритми інтеграції чат-бота для автоматизації внутрішніх процесів
4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд та аналіз використання чат-ботів
2. Дослідження сучасних технологій інтеграції та алгоритмізації чат-ботів для автоматизації внутрішніх процесів
3. Розробка алгоритму та програмна інтеграція чат-бота для автоматизаціх внутрішніх процесів
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
1. Загальна схема роботи чат-бота (рис. 1.2, ст. 20)
2. Список стоп-слів для англійської мови (рис. 22, ст. 36)
3. Приклад роботи Word2vec (рис. 2.3, ст. 38)
4. Високорівнева структурна схема BERT (рис. 2.4, ст. 39)
5. Приклад роботи OAuth (рис. 2.6, ст. 45)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	20.09.2024	виконано
2	Аналіз використання чат-ботів	01.10.2024	виконано
3	Способи забезпечення безпечної передачі даних у мережі інтернет	12.10.2024	виконано
4	Дослідження алгоритму інтеграції чат-бота	25.10.20234	виконано
5	Формулювання вимог та алгоритмів функціонування системи	05.11.2024	виконано
6	Програмна реалізація рішення	22.11.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 84 с., 22 рис., 1 табл., 41 джерел.

Тема: Моделі, методи та алгоритми інтеграції чат-ботів для автоматизації внутрішніх процесів в компаніях.

Об'єкт дослідження: моделі та алгоритми автоматизації внутрішніх процесів у компаніях.

Мета роботи: дослідження та розробка моделей, методів та алгоритмів для автоматизації внутрішніх процесів.

Предмет дослідження: технології та інструменти інтеграції чат-ботів у корпоративні інформаційні системи, а також методи автоматизації внутрішніх процесів компаній за допомогою чат-ботів.

Результати дослідження: Виконано аналіз існуючих рішень інтеграції чат-ботів у корпоративні системи, зокрема CRM, ERP та HRM. На основі отриманих результатів запропоновано нову архітектуру та алгоритми роботи чат-ботів, які забезпечують автоматизацію рутинних завдань, підвищення продуктивності співробітників і зменшення адміністративних витрат.

Висновок:

У результаті проведених досліджень створено ефективну систему інтеграції чат-ботів, яка дозволяє автоматизувати внутрішні бізнес-процеси. Використання сучасних методів обробки природної мови, алгоритмів управління діалогами та безпечної передачі даних забезпечує високу якість роботи системи.

АВТОМАТИЗАЦІЯ, ЧАТ-БОТИ, КОРПОРАТИВНІ СИСТЕМИ, ОБРОБКА ПРИРОДНОЇ МОВИ, API, БЕЗПЕКА ДАНИХ, ІНТЕГРАЦІЯ.

ANNOTATION

Master's work: 84 p., 22 fig., 1 tab., 41 sources.

Topic: Models, methods and algorithms for integrating chatbots to automate internal processes in companies.

Object of research: models and algorithms for automating internal processes in companies.

Purpose: research and development of models, methods, and algorithms for automating internal processes.

Subject of research: technologies and tools for integrating chatbots into corporate information systems, as well as methods for automating internal processes of companies using chatbots.

Research results: An analysis of existing solutions for integrating chatbots into corporate systems, including CRM, ERP, and HRM, was performed. Based on the results obtained, a new architecture and algorithms for chatbots were proposed to automate routine tasks, increase employee productivity, and reduce administrative costs.

Conclusion:

As a result of the research, an effective chatbot integration system has been created that allows automating internal business processes. The use of modern methods of natural language processing, dialog management algorithms, and secure data transfer ensures high quality of the system.

AUTOMATION, CHATBOTS, CORPORATE SYSTEMS, NATURAL LANGUAGE PROCESSING, API, DATA SECURITY, INTEGRATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API - прикладний програмний інтерфейс

SDK - набір із засобів розробки

NLP - обробка природної мови (Natural Language Processing)

ERP - система планування ресурсів підприємства (Enterprise Resource Planning)

CRM - система управління взаємовідносинами з клієнтами (Customer Relationship Management)

TLS - Transport Layer Security, протокол захищеної передачі даних

HTTPS - протокол безпечної передачі даних Hypertext Transfer Protocol Secure

RSA - криптографічний алгоритм із відкритим ключем (Rivest–Shamir–Adleman)

AES - симетричний алгоритм блочного шифрування (Advanced Encryption Standard)

AI - штучний інтелект (Artificial Intelligence)

ML - машинне навчання (Machine Learning)

API Gateway - шлюз прикладних програмних інтерфейсів

FNR - частота помилкових неспівпадінь (False Non-Match Rate)

JWT - токен для безпечної передачі інформації (JSON Web Token)

ВСТУП

Актуальність роботи

За останні роки внутрішні бізнес-процеси компаній ускладнилися через зростаючу потребу у швидкій обробці інформації, ефективній комунікації між співробітниками та автоматизації повторюваних операцій. Компанії, які здійснюють цифрову трансформацію бізнесу, повинні використовувати сучасні технології, щоб підтримувати економічну ефективність, конкурентоспроможність та результативність.

Чат-боти - це передова технологія автоматизації, яка вже продемонструвала свою перспективність у таких сферах зовнішньої діяльності, як маркетинг, продажі та обслуговування клієнтів. Вони все ще перебувають на ранніх стадіях реалізації свого потенціалу у внутрішній автоматизації, включаючи управління персоналом, документообіг, технічну підтримку та взаємодію з корпоративними системами.

Інтеграція чат-ботів з ERP-, CRM- та HRM-системами створює нові можливості для бізнесу, але водночас ставить перед ним низку викликів, зокрема необхідність забезпечення точності обробки природної мови (NLP), розробки універсальних моделей для ефективної взаємодії користувача з ботом та впровадження безпечної передачі даних.

Порівняння роботи з відомими розв’язаннями проблеми

Сучасні рішення з інтеграції чат-ботів для автоматизації бізнес-процесів демонструють різноманітність методів, які зосереджуються на різних аспектах інтеграції та функціонування. Однак вони мають переваги та недоліки, які послужили основою для розробки цієї статті.

Такі відомі платформи, як Microsoft Bot Framework, IBM Watson Assistant та Dialogflow, надають інструменти для розробки та впровадження чат-ботів, включаючи взаємодію з ERP, CRM та іншими бізнес-системами. Використовуючи НЛП і технології машинного навчання, вони забезпечують відмінну точність

розпізнавання природної мови. Однак через обмежену підтримку перекладу або потребу в додаткових API ці рішення можуть бути складними для інтеграції в корпоративний контекст, їх часто важко налаштувати, а також вони вимагають значних ресурсів для навчання моделей.

Інші системи, зокрема Facebook Messenger Platform і Telegram Bot API, пропонують низьку вартість встановлення та спрощений дизайн, але їхня функціональність здебільшого обмежена. Рідко коли ці рішення взаємодіють з корпоративними системами без необхідності спеціальних модифікацій або додаткових модулів.

Мета і задачі дослідження

Метою магістерської роботи є дослідження та розробка моделей, методів і алгоритмів інтеграції чат-ботів для автоматизації внутрішніх бізнес-процесів компаній.

Досліджувана модель інтеграції чат-ботів повинна забезпечувати ефективну взаємодію між користувачами та компонентами корпоративних систем за допомогою мережі Інтернет. Основною вимогою є можливість масштабування архітектури для підтримки великої кількості одночасних запитів, а також адаптація до використання на пристроях з обмеженими обчислювальними ресурсами, таких як мобільні телефони чи планшети.

Досягнення мети включало розв’язання таких **задач**:

1) огляд існуючих методів використання чат-ботів для автоматизації бізнес-процесів;

2) дослідження засобів інтеграції чат-ботів із корпоративними системами, такими як CRM, ERP та HRM;

3) аналіз алгоритмів обробки природної мови (NLP);

4) вибір архітектури інтеграції чат-бота, що враховує вимоги та обґрунтування доцільності його використання;

5) програмна реалізація прототипу чат-бота для автоматизації внутрішнього бізнес-процесу.

Об’єктом дослідження є моделі та алгоритми автоматизації внутрішніх процесів у компаніях. Предметом дослідження виступають технології та інструменти інтеграції чат-ботів у корпоративні інформаційні системи.

Предметом дослідження інформаційні технології інтеграції чат-ботів у корпоративні системи, алгоритми обробки природної мови (NLP), а також методи забезпечення безпеки передачі даних та взаємодії між компонентами бізнес-архітектури.

Методи дослідження: Для досягнення поставлених цілей у роботі використано методичний підхід для розробки архітектури, моделювання для створення алгоритмів управління діалогами та маршрутизації запитів, експеримент для тестування прототипу в реальних умовах, а також аналіз і синтез для вивчення існуючих підходів до інтеграції чат-ботів.

Наукова новизна одержаних результатів

Здійснено аналіз існуючих рішень для інтеграції чат-ботів із корпоративними системами, таких як CRM, ERP та HRM, а також методів обробки природної мови та забезпечення безпеки передачі даних. На основі проведеного дослідження запропоновано нову архітектуру та алгоритми роботи чат-бота, які оптимізують взаємодію з бізнес-системами, забезпечують автоматизацію рутинних процесів і підвищують загальну ефективність внутрішніх бізнес-процедур.

Практичне значення одержаних результатів.

На основі проведеного дослідження було розроблено функціонуючу систему інтеграції чат-ботів для автоматизації внутрішніх бізнес-процесів, яка може бути

впроваджена в існуючі корпоративні системи, такі як CRM, ERP та HRM. Створений прототип забезпечує автоматизацію рутинних завдань, підвищення продуктивності праці співробітників і покращення комунікації між користувачами та бізнес-системами, сприяючи оптимізації управління внутрішніми процесами компанії.

Особистий внесок

1. Виконано аналіз існуючих рішень інтеграції чат-ботів у корпоративні інформаційні системи (CRM, ERP, HRM).
2. Виявлено недоліки пов’язані з автоматизацією внутрішніх бізнес-процесів за допомогою чат-ботів.
3. Запропоновано архітектуру, алгоритм функціонування та реалізацію системи інтеграції чат-ботів для автоматизації внутрішніх процесів у компаніях із використанням сучасних методів обробки природної мови та забезпеченням високого рівня безпеки передачі даних через мережу Інтернет.

Структура магістерської роботи.

Магістерська робота викладена на 84 сторінках друкованого тексту, який складається з вступу, трьох розділів, висновків, списку використаних джерел (41 найменувань). Робота містить 22 рисунків.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ВИКОРИСТАННЯ ЧАТ-БОТІВ

Однією з найпоширеніших та найуспішніших технологій для спілкування бізнесу зі споживачами на даний момент є чат-боти. Вони дають змогу автоматизувати процедури обслуговування, що підвищує продуктивність і задоволеність клієнтів. Розвиток технологій машинного навчання та штучного інтелекту створює нові можливості для підвищення ефективності та якості роботи чат-ботів. Як показує статистика (рис.1.1.), чат-боти швидкими темпами стають все більш популярнішими.

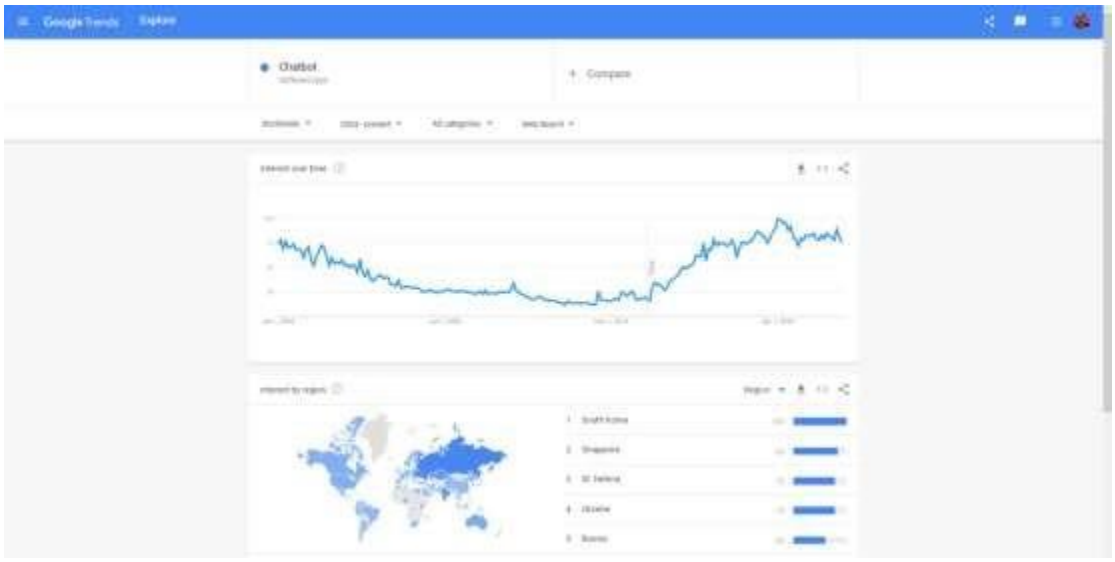

Рис. 1.1. Статистика популярності чат-ботів у світі

Метою цієї магістерської роботи є вивчення ринку бізнес-ботів, детальний аналіз технологій та процесів, що використовуються для їх створення, а також дослідження ключових підходів до їх розробки та впровадження. Крім того, важливим завданням є створення робочого прототипу системи розробки чат-ботів на основі штучного інтелекту, який дозволить продемонструвати практичне застосування отриманих теоретичних знань.

1.1 Історія і розвиток чат-ботів

1.1.1 Витоки та ранній розвиток чат-ботів

Створення ELIZA [2] в 1960-х роках ознаменувало початок пригод чат-ботів. ELIZA примітна тим, що є одним з перших і найбільш значущих чат-ботів. Її створив у середині 1960-х років Джозеф Вайзенбаум у Массачусетському технологічному інституті (MIT). Він мав імітувати розмови, подібні до розмов з роджеріанським психотерапевтом.

Щоб реагувати на введення користувача, Еліза використовувала прості алгоритми зіставлення шаблонів. Відповіді, які вона надавала, були терапевтичними і часто переформульовували репліки користувача у вигляді запитів. На той час вербальні навички Елізи були досить примітивними. Їй також потрібно було попрацювати над вербальним розумінням. Однак це, безсумнівно, показало, як комп'ютерні програми можуть спілкуватися з людьми.

Після ELIZA значним досягненням стала програма PARRY [3] Кеннета Колбі, розроблена в 1972 році. Це програмне забезпечення імітувало дії пацієнта з параноїдальною шизофренією. Для того, щоб виробляти психологічно реалістичні відповіді, вона базувалася на більш складних моделях, які містили евристичні правила. PARRY був першим «мислячим» ботом, оскільки він використовував моделювання емоційного стану, на відміну від ELIZA, який відповідав лише на певні ключові слова.

У серії тестів PARRY піддавали випробуванням: психіатри взаємодіяли з програмою, не усвідомлюючи, що це був бот. Вони помилково вважали, що відповіді програми були справжньою людською мовою, що доводило її реалістичність.

В ігровій індустрії чат-боти мають важливе значення для підвищення взаємодії та залучення гравців. Їх можна використовувати для створення інтерактивних NPC (неігрових персонажів), автоматизації вирішення технічних проблем, налаштування ігрового досвіду та підтримки користувачів. Використовуючи реалістичну мову, ці

боти посилюють занурення гравців у розповідь і дають їм негайні відповіді на запитання, не вимагаючи від них виходу з гри.

Штучний інтелект і обробка природної мови дозволили чат-ботам навчати новачків, надавати поради та автоматично збирати інформацію від розробників, щоб швидко знаходити і виправляти проблеми. Крім того, чат-боти часто використовуються для створення ігрових спільнот, аналізу поведінки гравців і таргетованої реклами.

1.1.2 Еволюція функціональності чат-ботів до 2010-тих

Хоча розробка чат-ботів залишалася відносно обмеженою до 2010 року, вона заклала основу для майбутніх застосувань штучного інтелекту. Можливості цих систем значно зросли з розвитком масивних мовних моделей і комп'ютерних потужностей.

Чат-боти почали інтегруватися з базами знань у 1980-х роках, коли стали доступними більш потужні обчислювальні ресурси. Наприклад:

- Одним із перших ботів, який створив зв'язний текст, що нагадував художні твори, зокрема короткі романи, був Racter (1983).
- У цей час також більше уваги приділялося моделям генерації мови, які генерували відповіді за допомогою випадкових алгоритмів.
- У 1995 році було створено бота на ім'я Jabberwacky, який мав на меті імітувати реалістичний людський діалог. З часом ця програма змогла покращити свої відповіді, зберігаючи історію минулих дискусій.
- Почали з'являтися боти для корпоративного використання. Наприклад, контакт-центри обробляли запити за допомогою перших систем автоматизації обслуговування клієнтів.

Чат-боти почали впроваджуватися в AOL Instant Messenger та MSN Messenger на початку 2000-х років. **S**marterChild та інші боти для прогнозу погоди, новин та розваг набули популярності. Хоча ці боти могли швидко надавати інформацію, їм бракувало навичок самонавчання.

Ключові інновації цього періоду:

- Зростання обсягу даних, доступних для аналізу ботами.
- Використання більш просунутих алгоритмів обробки мови для покращення точності діалогів.

1.1.3 Сучасна епоха чат-ботів (2010-ті роки)

Завдяки появі штучного інтелекту (ШІ), обробці природної мови (NLP) та зростанню кількості додатків для обміну повідомленнями, таких як Facebook Messenger, Telegram та Slack, розробка чат-ботів була особливо активною у 2010-х роках. Чат-боти, які ефективно спілкуються з користувачами за допомогою текстового або голосового інтерфейсу, активно використовуються як в особистих, так і в професійних цілях.

Роль соціальних мереж у розвитку ботів

Facebook Messenger запустив API для чат-ботів у 2016 році, що сприяло зростанню інтерактивних компаній. У 2015 році Telegram застосував схожий підхід, запропонувавши платформу для створення ботів з розширеними можливостями, включаючи обробку платежів, автоматизацію чатів та взаємодію зі спільнотами. Завдяки своїй здатності інтегрувати ботів для підвищення продуктивності та управління завданнями, Slack набув популярності серед бізнес-команд. Ці платформи стають важливими бізнес-інструментами, які дозволяють швидко комунікувати з клієнтами та співробітниками.

Інтеграція чат-ботів зі штучним інтелектом

Розвиток NLP і машинного навчання покращив розуміння чат-ботами людської мови та їхню здатність контекстуалізувати свої відповіді. Наприклад, чат-боти, які використовують моделі на кшталт GPT (включно з новітніми рішеннями OpenAI), здатні створювати складні, органічні відповіді. Ці боти використовуються в управлінні даними, підтримці клієнтів, індивідуальних пропозиціях та інших сферах.

Застосування голосових ботів

Увімкнувши мовну взаємодію, голосові помічники, такі як Siri (2010), Google Assistant (2012) та Amazon Alexa (2015), додали новий рівень взаємодії.

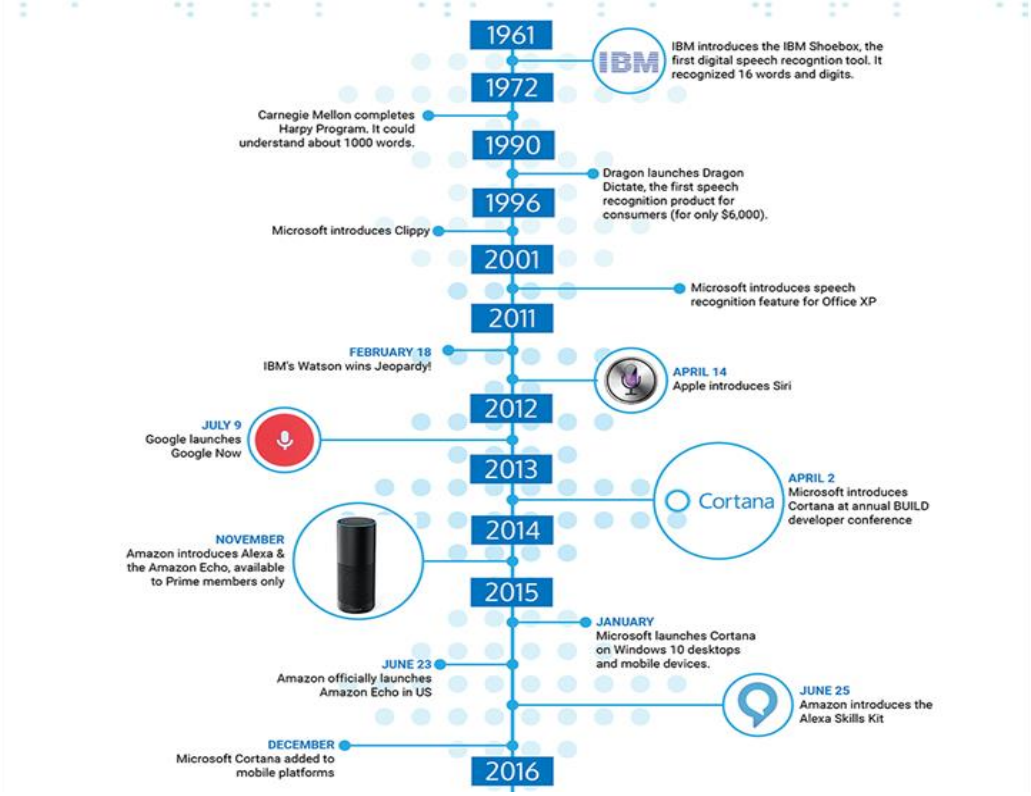

Рис. 1.2. Етапи розвитку голосових чат-ботів

Google Assistant розширив пошукові та контекстні можливості, Alexa впровадила голосове управління в пристрої «розумного будинку», а Siri стала першою голосовою системою, вбудованою в мобільний пристрій. На рисунку 1.2 зображено етапи розвитку голосових чат-ботів . Ці системи спростили доступ до інформації та управління завданнями у повсякденному житті.

1.1.4 Чат-боти в сучасному світі

Чат-боти стають важливим компонентом цифрової екосистеми, впливаючи на автоматизацію процесів, обслуговування клієнтів та бізнес. Завдяки широкій доступності платформ, простоті побудови та взаємодії зі штучним інтелектом їхнє використання розширюється.

Стан ринку та популярні платформи

- *Dialogflow [4]*: Платформа від Google з інтуїтивно зрозумілим інтерфейсом і підтримкою багатоканальної інтеграції.
- *Microsoft Bot Framework [5]:* Фреймворк для створення ботів з інтеграцією в Azure і підтримкою мультиканальності.
- *Amazon Lex [6]:* Забезпечує розробку голосових і текстових ботів з використанням технології AWS, яка також використовується для Amazon Alexa.
- *Rasa:* Відкритий фреймворк для розробників, які потребують високого рівня кастомізації.
- *IBM Watson Assistant [7]:* Забезпечує надійність і масштабованість для великих компаній.

Індустрії, що активно використовують ботів

- *Медицина:* Боти використовуються для запису на прийоми, відповіді на поширені запитання пацієнтів і моніторингу стану здоров'я.
- *Фінанси:* Вони допомагають з обробкою запитів, управлінням рахунками та консультаціями з фінансових питань.
- *Роздрібна торгівля:* Боти оптимізують процеси обслуговування клієнтів, підтримують онлайн-продажі та покращують клієнтський досвід.

Виклики та обмеження

- *Етичні аспекти*: Питання прозорості та використання даних залишається ключовим викликом.
- *Приватність*: Забезпечення захисту персональних даних є важливою вимогою.
- *Безпека:* Захист від витоків даних і кібератак є важливим для забезпечення довіри користувачів.

1.2 Теоретичні відомості про чат-боти

1.2.1 Визначення та класифікація чат-бота

Чат-боти – це програми, здатні імітувати спілкування з користувачем на

природній мові. Джозеф Вайценбаум створив першого чат-бота ELIZA у 1966 році, що стало початком їхнього розвитку.

Для того, щоб знайти ключові слова та фрази для відповідей на запити користувачів, ELIZA використовувала базові шаблони. З тих пір технологія чат-ботів пройшла довгий шлях завдяки прогресу в обробці природної мови та машинному навчанню, що значно змінило схему її роботи(рис. 1.3.), збільшивши їхні можливості.

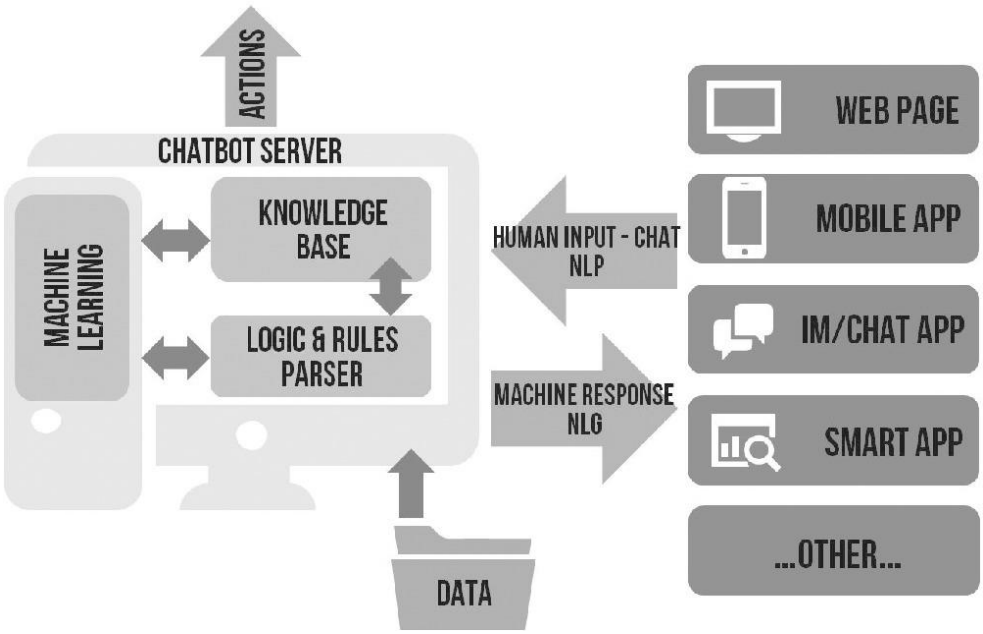

Рис. 1.3. Загальна схема роботи чат-бота

Чат-бот - це комп'ютерна програма, створена для спілкування з користувачами за допомогою голосових або текстових повідомлень. В основі чат-бота лежить набір алгоритмів, які оцінюють вхідні дані користувача та надають відповідь. Чат-боти можна класифікувати на основі низки факторів, таких як технологія, яку вони використовують, сфера їх застосування та спосіб, у який вони ведуть розмову.

Можна виділити такі основні сфери застосування:

1. Обслуговування клієнтів:

- Підтримка клієнтів: Надає інформацію та допомогу у вирішенні питань.
- FAQ: Легкий та швидкий доступ до відповідей на найпоширеніші

запитання.

2. E-commerce та рітейл:

- Рекомендації товарів: Підбір продуктів на основі запитів користувачів в електронній комерції та роздрібній торгівл.
- Спрощення процесу покупки: Допомога в оформленні замовлення, відстеженні доставки.

3. Фінансовий сектор:

- Інформація про рахунки: Баланс, історія транзакцій та інформація про рахунки.
- Фінансові консультації: Пропозиції щодо кредитування, інвестування тощо.

4. Медицина та здоров'я:

- Поради щодо здоров'я: Відповіді на запити щодо симптомів і методів лікування.
- Нагадування про прийом ліків.

5. Освіта:

- Допомога з домашнім завданням і відповіді на педагогічні запити - приклади того, як можна підтримати навчання.
- Інтерактивне навчання: Використання чат-ботів для створення інтерактивних уроків.

6. Туризм та гостинність:

- Бронювання житла, подорожей та екскурсій.
- Інформаційна підтримка: пропозиції щодо напрямків та деталей про регіональні традиції.

7. Розваги та медіа:

- Пропонований контент включає книги, музику та фільми.
- Чат-боти використовуються як персонажі в інтерактивних іграх.

8. Внутрішні корпоративні потреби:

- HR-підтримка: Підтримка персоналу включає в себе навчання,

опитування задоволеності працівників та відповіді на їхні запити.

- IT-підтримка: включає навчання персоналу та допомогу з розв’язанням технологічних завдань.

9. Маркетинг та продажі:

- Індивідуальні маркетингові кампанії: Залучення чат-ботів для спілкування з потенційними клієнтами.
- Зворотній зв’язок з користувачами: Опитування споживачів про товари та послуги.

На рисунку 1.4 продемонстровано приклад боту підтримки клієнтів Yasno який допомагає користувачам передавати показники лічильника, з сплатою рахунків та нагадує користувачам про подання показників, що дозволяє клієнтам даного сервісу не пропускати потрібні платежі і вчасно сплачувати за комунальні послуги.

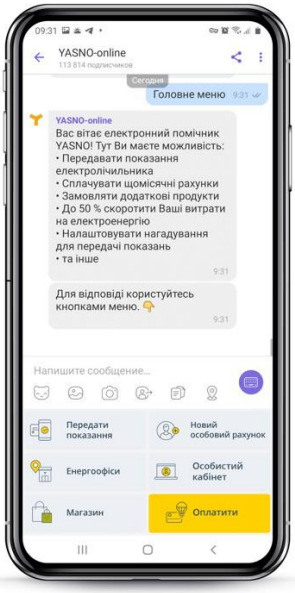

Рис. 1.4. Приклад боту який помагає користувачам сервісу Yasno

Чат-боти продовжують розвиватися, і ми можемо очікувати, що в майбутньому вони будуть використовуватися ще ширше, особливо в сферах, пов'язаних з обробкою великих обсягів даних, машинним навчанням і штучним інтелектом.

1.2.2 Приклади використання чат-ботів

Сценарні чат-боти

Обслуговування клієнтів в електронній комерції: Чат-боти на веб-сайтах електронної комерції допомагають клієнтам знайти товар, відповісти на поширені запитання і навіть оформити замовлення.

Банківська справа: Багато банків використовують чат-ботів для оптимізації певних процесів, таких як переказ грошей, перевірка балансу та вирішення проблем клієнтів.

Чат-боти на базі штучного інтелекту

Напевно, найвідомішим видом чат-ботів є віртуальний асистент. Чат-боти, такі як Siri від Apple , Google Assistant та Alexa від Amazon , можуть виконувати широкий спектр завдань - від надсилання текстових повідомлень і встановлення нагадувань до керування «розумним» будинком і відтворення музики.

Чат-боти для переговорів: призначені для ведення складних ситуативних переговорів і прийняття рішень. У корпоративному світі вони можуть допомогти в переговорах, пропонуючи пропозиції та аналізуючи потенційні результати.

Освітні чат-боти

Duolingo: Ця популярна програма для вивчення мови допомагає користувачам краще вивчити мову, використовуючи чат-ботів для імітації реальних сценаріїв спілкування.

Чат-боти для вступу до коледжів: Кілька університетів використовують чат-ботів, щоб допомогти абітурієнтам у процесі подання заяв, відповідаючи на їхні запити цілодобово.

Розважальні чат-боти

Mitsuku [9] - кумедний чат-бот, який може спілкуватися з людьми на різні теми, складаючи їм компанію.

Чат-боти для ігор: Багато мобільних додатків та відеоігор використовують чат-ботів для створення цікавого інтерактивного досвіду.

Ці ілюстрації підкреслюють різноманітність використання чат-ботів і показують, наскільки корисними вони можуть бути для цілого ряду галузей.

Важливо також пам'ятати, що з розвитком технологій чат-боти стають розумнішими і здатними до складних діалогів, що створює для них нові сфери застосування.

1.2.3 Огляд платформ для створення чат-ботів

Дуже важливо вибрати найкращу платформу для розробки чат-ботів, яка б задовольняла всі специфікації та потреби проекту. Існує багато платформ для розробки чат-ботів, і кожна з них має свої особливості, переваги та недоліки. Короткий огляд деяких з найбільш поширених платформ можна побачити нижче.

Таблиця 1.1

Огляд платформ які використовують чат-боти

Платформа	Переваги	Недоліки	Технічні деталі
Dialogflow від Google	Інтеграція з іншими продуктами Google; Потужні можливості обробки природної мови; Можливість використання на різних платформах.	Складнасистема для початківців; Обмеження у безкоштовній версії.	Використовує технології машинного навчання для розуміння намірів користувача та витягування ентіті.
Microsoft Bot Framework	Велика спільнота та підтримка; Інтеграція з Azure та іншими сервісами Microsoft;	Складна система для початківців; Висока залежність від сервісів Microsoft	Комплексний набір інструментів для створення чат- ботів, включаючи SDK та служби для розробників

Продовження таблиці 1.1

IBM Watson Assistant	Високий рівень розуміння природної мови; Можливість створення складних діалогових систем;	Висока ціна; Складність налаштування і використання.	Використовує потужні технології штучного інтелекту IBM для створення чат- ботів.
Telegram Bot API [9]	Легкість використання; Велика кількість користувачів Telegram; Безкоштовність.	Обмежені можливості (в порівнянні з іншими платформами); Відсутність вбудованих інструментів для обробки природної мови.	Платформа для створення ботів у месенджері Telegram. Для обробки природньої мови треба підключати додаткові технології.
Facebook Messenger Platform [10]	Величезна аудиторія Facebook Messenger; Інтеграція з Facebook Ads; Легкість розповсюдження бота.	Обмежені можливості взаємодії з користувачем (в порівнянні з іншими платформами); Залежність від політики Facebook.	Платформа для створення ботів у месенджері Facebook. Для обробки природньої мови треба підключати додаткові технології.

Залежно від потреб і завдань проекту, ці платформи дозволяють розробникам створювати чат-ботів з різним ступенем функціональності та складності. На вибір платформи впливають численні фактори, такі як особисті уподобання розробника, цільова аудиторія, необхідний функціонал і бюджет.

1.2.4 Основні функціональні можливості чат-ботів

У сучасному світі чат-боти можуть виконувати найрізноманітніші завдання, гарантуючи ефективну та зручну взаємодію з користувачами. Нижче наведено деякі з основних функцій, які вони надають:

1. *Обробка природної мови (NLP)[11]:* Чат-боти можуть розуміти наміри користувача і давати відповіді завдяки своїм складним навичкам аналізу та обробки природної мови. Це охоплює такі мовні функції, як визначення контексту та розпізнавання ключових слів**.**

2. *Ведення діалогу та контекст:* Багатосторонні розмови можуть підтримуватися чат-ботами, які можуть згадувати минулі обміни думками і враховувати їх у наступних комунікаціях. Це створює видимість змістовного та органічного дискурсу.

3. *Інтеграція з іншими системами:* Численні бази даних, API та зовнішні сервіси можна поєднати з чат-ботами, щоб надати їм доступ до великої кількості даних і дати можливість виконувати практичні завдання.

4. *Навчання та адаптація:* Навчаючись на минулому досвіді, сучасні чат-боти можуть краще розуміти запити користувачів і надавати більш точні відповіді.

5. *Автоматизація процесів:* Чат-боти необхідні для автоматизації багатьох операцій компанії, таких як обробка замовлень, взаємодія з клієнтами та надання інформації про товари та послуги.

6. *Аналітика та звітність:* Чат-боти можуть збирати та аналізувати великі обсяги даних про взаємодію з користувачами, надаючи організаціям корисну інформацію для прийняття рішень.

Завдяки цим можливостям чат-боти є ефективним інструментом, який допомагає організаціям скоротити витрати, поліпшити обслуговування клієнтів і підвищити продуктивність.

1.2.5 Використання штучного інтелекту в чат-ботах

Завдяки своєму проникненню в різні сфери та впливу на те, як ми живемо,

працюємо та розважаємося, прикладний штучний інтелект швидко стає необхідним компонентом повсякденного життя.

Чат-боти тепер можуть функціонувати і бути більш ефективними завдяки додаванню штучного інтелекту (ШІ), який робить їх розумнішими, більш адаптивними і здатними давати більш точні і корисні відповіді. Розглянемо застосування ШІ в чат-ботах більш детально:

1. Аналіз даних і машинне навчання: ШІ дозволяє чат-ботам отримувати знання з попередніх обмінів і поступово покращувати свої відповіді. Великі обсяги даних аналізуються алгоритмами машинного навчання, які знаходять закономірності в діалозі з користувачем і покращують процес прийняття рішень ботом.

2. Ключовим елементом технології, що дозволяє чат-ботам розуміти, оцінювати та створювати природну мову, є обробка природної мови (NLP). Використовуючи обробку природної мови (NLP), бот може краще розуміти наміри користувача і відповідати релевантною інформацією.

3. Машинний зір: Деякі чат-боти аналізують надіслані користувачем фотографії та відео за допомогою машинного зору. Це може бути корисно для різних цілей, наприклад, для ідентифікації товарів у роздрібній торгівлі або допомоги в діагностиці в медицині.

4. Глибоке навчання: Завдяки глибокому навчанню деякі з найбільш просунутих чат-ботів можуть обробляти величезні обсяги даних і виконувати складні завдання, такі як переклад тексту, розпізнавання звуку і навіть створення власних відповідей.

5. Підтримка прийняття рішень: ШІ дозволяє чат-ботам оцінювати обставини, обирати найкращий спосіб дій і пропонувати користувачам пропозиції на основі зібраних даних.

6. Персоналізація: ШІ дозволяє чат-ботам коригувати свої рекомендації та відповіді на основі попередніх взаємодій користувача та його особистих уподобань.

7. Автоматизація та оптимізація процесів: ШІ сприяє автоматизації широкого спектру завдань, включаючи обробку замовлень, відповіді на поширені

запитання і навіть вирішення складних проблем.

8. Постійне вдосконалення: Чат-боти зі штучним інтелектом здатні до самовдосконалення. Вони аналізують свої помилки, витягують з них уроки та вдосконалюють алгоритми, щоб надавати точніші відповіді в майбутньому.

Можливості чат-ботів значно розширюються, коли в них впроваджується штучний інтелект, що підвищує їхню здатність пропонувати користувачам кращу допомогу та послуги. Це покращує користувацький досвід, а також допомагає оптимізувати внутрішні бізнес-процедури.

Створення розумних, ефективних і практичних віртуальних помічників вимагає впровадження штучного інтелекту в чат-ботів. Завдяки штучному інтелекту чат-боти можуть самонавчатися, розуміти природну мову, оцінювати фотографії, приймати обґрунтовані рішення та налаштовувати відповіді. Разом ці фактори призводять до створення по-справжньому інтелектуальних систем, які можуть підлаштовуватися під запити користувачів і пропонувати їм найкращу можливу допомогу.

Однак важливо також пам'ятати, що інтеграція штучного інтелекту в чат-ботів вимагає значних витрат часу, грошей і залучення фахівців з машинного навчання та обробки природної мови. Тим не менш, витрати на цю технологію можуть значно підвищити якість обслуговування клієнтів, спростити внутрішні процедури та запропонувати більш глибокий і персоналізований користувацький досвід.

1.2.6 Аналіз ринку та перспектив розвитку чат-ботів

Світовий ринок чат-ботів швидко зростає, і, згідно з останніми дослідженнями, його обсяг може значно збільшитися протягом наступних кількох років. Компанії все частіше використовують чат-ботів у своїх планах, щоб покращити обслуговування клієнтів, скоротити витрати та спростити внутрішні процедури.

*Зростання популярності та інтеграція з месенджерами***.** З кожним роком все більше компаній бачать цінність включення чат-ботів у свої канали комунікації. Це пов'язано з тим, що використання месенджерів є простим і дозволяє користувачам

швидко і легко отримувати відповіді на свої запитання. Інтегруючи чат-ботів з такими відомими платформами, як Facebook Messenger, WhatsApp і Telegram, компанії можуть полегшити зв'язок зі своїми клієнтами.

Покращення технологій обробки природної мови (NLP). Технології обробки природної мови мають важливе значення для створення чат-ботів. Вони роблять взаємодію з чат-ботами більш ефективною і природною, дозволяючи машинам розуміти і обробляти людську мову. Постійний розвиток НЛП робить чат-ботів розумнішими та компетентнішими у виконанні дедалі складніших завдань.

*Підтримка та аналіз даних***.** Чат-боти пропонують потужні інструменти для аналізу даних та обслуговування клієнтів. Вони можуть автоматично відповідати на запити, пропонувати пропозиції та досліджувати величезні обсяги даних, щоб виявити закономірності та покращити підтримку клієнтів.

*Роль ШІ та машинного навчання***.** Розробка чат-ботів значною мірою спирається на машинне навчання та штучний інтелект. Вони дозволяють чат-ботам вчитися на минулому досвіді, підлаштовуватися під потреби користувачів і надавати більш індивідуалізовані відповіді на додаток до звичайних запитів.

*Майбутнє чат-ботів: інтеграція та інновації***.** На чат-ботів чекає світле майбутнє. Очікується, що вони й надалі інтегруватимуться з низкою корпоративних операцій, включаючи автоматизацію завдань, внутрішню аналітику та обслуговування клієнтів. Розвиток штучного інтелекту та машинного навчання продовжуватиме покращувати розуміння та аналіз людської мови чат-ботами, що дозволить їм надавати дедалі точніші та корисніші відповіді.

Глобальна конкуренція та локалізація Конкуренція між розробниками та постачальниками цих технологій зростає разом зі світовим ринком чат-ботів. Щоб отримати конкурентну перевагу, численні компанії спрямовують значні ресурси на створення чат-ботів. Чат-боти повинні бути одночасно локалізовані, щоб спілкуватися з користувачами з різним мовним і культурним досвідом. Щоб досягти максимального успіху, компанії повинні забезпечити своїх чат-ботів якісними перекладами та культурною адаптацією.

Етика та безпека Технології обробки природної мови мають важливе значення для створення чат-ботів. Вони роблять взаємодію з чат-ботами більш ефективною і природною, дозволяючи машинам розуміти і обробляти людську мову. Постійний розвиток НЛП робить чат-ботів розумнішими та компетентнішими у виконанні дедалі складніших завдань.

Соціальний вплив і прийняття технології оскільки чат-боти змінюють те, як люди використовують технології та отримують інформацію, вони також мають значний соціальний вплив. Їхнє широке використання може сприяти розвитку більш зручних та ефективних послуг, але також може викликати занепокоєння щодо втрати робочих місць у галузях, де людська праця може бути замінена роботами. При створенні та впровадженні чат-ботів дуже важливо враховувати ці соціальні фактори, щоб гарантувати всебічно обґрунтовану стратегію, яка сприятиме розвитку креативності та соціальної згуртованості.

Ринок чат-ботів загалом стрімко розширюється і розвивається, створюючи нові перспективи для компаній і змінюючи традиційні методи взаємодії з клієнтами. На додаток до технологічного прогресу, дуже важливо гарантувати безпеку та конфіденційність чат-ботів, беручи до уваги моральні та соціальні наслідки їхнього використання. На розробку чат-ботів чекає світле майбутнє, сповнене можливостей для інновацій та кращого обслуговування клієнтів.

1.2.7 Аналіз ринку та перспектив розвитку чат-ботів

При створенні та використанні чат-ботів необхідно враховувати безпеку даних та інформації користувачів. У сучасну цифрову епоху конфіденційність і безпека даних є головними проблемами.

Основні аспекти безпеки включають:

1. Шифрування даних: Кожна розмова користувача та частина приватної інформації повинна бути зашифрована. У випадку, якщо інформація буде перехоплена сторонніми особами, це гарантує її конфіденційність.

2. Автентифікація та авторизація: Щоб гарантувати, що тільки авторизовані

користувачі можуть отримати доступ до бота і його послуг, використовуються рішення для аутентифікації, такі як токени JWT.

3. Обмеження доступу до даних: надання обмеженого доступу до серверів і баз даних, в яких зберігаються дані користувачів.

4. Моніторинг та журналування: Відстеження активності для виявлення та усунення потенційних вторгнень або незаконного доступу.

5. Регулярне оновлення та патчінг: Слідкуйте за виправленнями та оновленнями для технологій та систем, які використовуються для усунення вразливостей.

6. Навчання користувачів: Навчання користувачів найкращим практикам безпеки, таким як уникнення використання ботів для передачі приватної інформації.

7. Резервне копіювання: Регулярне створення резервних копій даних допоможе відновити їх у разі втрати або пошкодження.

Проблеми конфіденційності:

1. Зберігання особистої інформації: Особиста інформація користувачів не повинна зберігатися без їхнього відома або згоди.

2. Використання данних: Потрібно пояснювати користувачам, як і для чого буде використовуватися їхня інформація.

3. Видалення даних: Надавати споживачам можливість видаляти їхні персональні дані з бази даних.

Безпека стає головною проблемою в результаті швидкого розвитку технологій і зростаючого обсягу даних, що обробляються чат-ботами. На додаток до захисту користувачів, належна стратегія захисту даних і конфіденційності може підвищити довіру до чат-ботів та їхніх пропозицій.

1.3. Висновки до розділу

В даному розділі було згадано про розробку таких систем, як ELIZA та PARRY, яка продемонструвала початковий потенціал текстової людино-машинної

взаємодії. Ці системи показали, як можна імітувати людське спілкування, і заклали основу для подальших досліджень, будучи при цьому простими і заснованими на попередньо встановлених шаблонах. Вони не лише демонстрували можливість створення діалогу між людиною і машиною, але й стали важливою відправною точкою для розвитку сучасних технологій обробки природної мови.

Використання чат-ботів зросло у 1980-х роках завдяки інтеграції баз знань та алгоритмів генерації мовлення, що дозволило генерувати більш зрозумілі тексти. Це було особливо актуально в ігровому секторі та творчих програмах, таких як програма Racter. Хоча боти все ще покладалися на заздалегідь написані сценарії, вони вже демонстрували здатність до певної імітації інтелектуальної взаємодії. Перехід до інтерактивного спілкування в 1990-х роках на прикладі Jabberwacky зробив розмови з ботами більш природними, дозволяючи користувачам відчувати себе у діалозі з реальною особистістю.

Взаємодіючи з мережами обміну повідомленнями (AOL, MSN), боти на кшталт SmarterChild вже на початку 2000-х років пропонували корисну інформацію та розширені можливості для користувачів. За цей час обсяги даних збільшилися, а алгоритми обробки природної мови вдосконалилися, що дозволило ботам надавати відповіді швидше та точніше. Однак повноцінних функцій самонавчання все ще було мало, а більшість відповідей залишалися залежними від заздалегідь створених шаблонів.

До 2010 року чат-боти пройшли значний шлях від скриптових програм до інтерактивних інструментів, які стали невід'ємною частиною повсякденного життя, розваг та бізнесу. Пропонуючи платформу для застосування глибокого навчання, штучного інтелекту та взаємодії з кількома сервісами, вони створили основу для сучасних технологій. Цей період став переломним для індустрії, оскільки розробники отримали доступ до більш потужних інструментів та алгоритмів, що забезпечили підвищену функціональність та універсальність ботів.

Платформи, які дозволяють створювати та розгортати чат-ботів, отримали особливу увагу в процесі розробки. Такі платформи, як IBM Watson Assistant,

Microsoft Bot Framework, Google Dialogflow та Telegram Bot API, надали розробникам доступ до надійних інструментів для інтеграції та персоналізації чат-ботів. Вибір платформи базується на технічних вимогах і цілях проекту. Наприклад, Telegram Bot API зручний для швидкого виправлення помилок та налаштувань, а Microsoft Bot Framework пропонує універсальність у створенні ботів для різних каналів зв'язку та інтеграцій.

Таким чином, технології та досвід, набуті до 2010 року, забезпечили міцний фундамент для розробки більш функціональних та інтерактивних систем у наступні роки. Ці досягнення зараз мають вирішальне значення для цифрової революції в бізнесі, оскільки чат-боти стають потужним інструментом для автоматизації та оптимізації процесів. Вони продовжують активно розвиватися, відкриваючи нові можливості для бізнесу, обслуговування клієнтів та інтеграції з іншими цифровими платформами. Подальші дослідження у цій сфері спрямовані на вдосконалення алгоритмів штучного інтелекту та забезпечення ще більш природної взаємодії між користувачами та системами. Інтеграція чат-ботів у різні галузі продовжує зростати, демонструючи їхню важливість у сучасному технологічному ландшафті. Очікується, що в майбутньому чат-боти стануть ще більш адаптивними до індивідуальних потреб користувачів. Розробники активно працюють над зменшенням помилок у діалогах і підвищенням якості мовного розпізнавання. Застосування чат-ботів у медицині, освіті та державному управлінні обіцяє значні покращення в ефективності послуг. Ринок чат-ботів продовжує демонструвати стійке зростання, що підтверджує їхній потенціал у цифровій економіці. З кожним роком технології стають доступнішими для малого та середнього бізнесу, що сприяє їх швидкому впровадженню.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ІНТЕГРАЦІЇ ТА АЛГОРИТМІЗАЦІЇ ЧАС-БОТІВ ДЛЯ АВТОМАТИЗАЦІЇ ВНУТРІШНІХ ПРОЦЕСІВ

2.1 Методи обробки природної мови для аналізу запитів

2.1.1 Методи попередньої обробки тексту: токенізація, видалення стоп-слів, лематизація та стемінг

Попередня обробка тексту є важливим етапом обробки природної мови (NLP), який готує текстові дані для подальшого аналізу.

Його мета — перетворити неструктурований текст у структурований формат, який можна аналізувати за допомогою алгоритмів машинного навчання та інших методів.

Основними методами попередньої обробки є токенізація, видалення стоп-слова, лемматизація та формування основи.

Кожен із цих підходів виконує свою функцію, але має переваги та недоліки.

Токенізація – це процес, за допомогою якого велика кількість тексту ділиться на менші частини, які називаються токенами. Ці лексеми дуже корисні для пошуку шаблонів і вважаються базовим кроком для створення коренів і лемматизації. Токенізація також допомагає замінити конфіденційні елементи даних неконфіденційними [13].

Обробка природної мови використовується для створення таких програм, як класифікація тексту, інтелектуальні чат-боти, аналіз настроїв, переклад мови тощо. Для досягнення вищезазначеної мети необхідно розуміння структури тексту.

Розгляньте наведений нижче приклад (рис. 2.1) використання токенізатора з бібліотеки *NLTK*, щоб зрозуміти різницю між токенізацією речень і токенізацією слів.

```
from nltk.tokenize import sent_tokenize
text = "God is Great! I won a lottery."
print(sent_tokenize(text))

Output: ['God is Great!', 'I won a lottery ']
```

Рис. 2.1. Приклад токенізера NLTK

Цей процес можна вважати однією з підзадач завдання аналізу вхідних даних. Оскільки українська мова суттєво не відрізняється від мов, наприклад, англійської щодо творення слів (слова, що складаються з літер, розділових знаків не відрізняються від подібних слів англійської та більшості європейських мов), то лексема не має певні характеристики.

Важливі особливості, властиві його застосуванню до письма, було складено укр.

Видалення стоп слів. Ці слова відфільтровуються перед будь-яким завданням попередньої обробки або моделювання. Стоп-слова обираються з огляду на їх незначущисть для поточного завдання NLP. Наприклад, список англійських стоп-слів у пакеті nltk визначає для виключення такі загальні слова, як-то *a, to, can* [14]. Більш обширний список стоп-слів зображений нижче (рис. 2.2)

Якщо перед нами стоїть завдання класифікації тексту чи аналізу настроїв, ми повинні видалити стоп-слова, оскільки вони не надають жодної інформації для нашої моделі, тобто видалити небажані слова з нашого корпусу, але якщо у нас є завдання перекладу мов, стоп-слова є корисні, оскільки їх потрібно перекладати разом із рештою слів.

Немає жорсткого правила щодо того, коли потрібно видаляти стоп-слова.

Але було б краще видалити стоп-слова, якщо наше завдання передбачає класифікацію мови, фільтрацію спаму, створення субтитрів, автоматичне створення тегів, аналіз настроїв або будь-що, пов’язане з аналізом типу тексту.

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were',
'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because',
'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after',
'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there',
'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only',
'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will',
'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm',
'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',
"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

Рис. 2.2. Список стоп-слів для англійської мови [15]

З іншого боку, якщо нашим завданням є машинний переклад, вирішення проблем питань і відповідей, узагальнення тексту, моделювання мови краще не видаляти слова stop, оскільки вони є важливою частиною цих програм.

Видалити стоп-слова за допомогою бібліотеки Python досить просто, і є різні способи зробити це.

Набір інструментів природної мови, або частіше NLTK, — це набір бібліотек і програм для символічної та статистичної обробки природної мови для англійської мови, написаних мовою програмування Python.

Він містить бібліотеки обробки тексту для кодування, аналізу, сортування, вкорінення, розмітки та семантичного обґрунтування.

Лематизація та стемінг.

Лематизація передбачає трансформацію форми слів у первісну. Це може бути іменник або прикметник в однині та називному відмінку. При цьому прикметник має бути у чоловічому роді. Якщо лема створюється з дієслова, він буде в інфінітиві [16].

На відміну від стемінгу, лематизація враховує граматичну роль слова та його значення в контексті речення. Цей підхід дозволяє проводити аналіз не лише окремого слова, а й ширшого контексту, наприклад, у межах сусідніх речень чи навіть у межах усього документа. Через це створення ефективних алгоритмів для лематизації є актуальною та складною задачею в сучасних дослідженнях.

Лематизація тісно пов’язана зі стемінгом, але між ними є важлива різниця.

Один із найпростіших способів реалізації лематизації — це пошук у словнику, що працює добре для базових форм. Однак для складних мов із довгими словами може знадобитися система, заснована на правилах, які створюються вручну або автоматично на основі текстового корпусу з анотаціями.

Стемінг – це знаходження стеми слова (основи). При перетворенні вам потрібно буде використовувати основне лексичне значення вибраного терміна. Найчастіше достатньо відкинути деякі частини (закінчення, суфікс). У російській мові будова слів набагато складніша, ніж у англійській, тому додатково можна використовувати лематизацію та інші алгоритми [18].

Стемінг обробляє кожне слово окремо, не враховуючи контексту, що призводить до однакового результату для слів, які мають різні значення залежно від ситуації. Водночас стемери простіші в реалізації та працюють значно швидше. Їх використання може бути виправданим у випадках, коли точність не є критичною.

Однак, у складних завданнях, де важливим є розуміння контексту, рекомендується використовувати лематизацію або більш просунуті методи обробки тексту.

Наприклад, у системах пошуку інформації стемінг часто допомагає досягти кращої швидкості обробки або навіть підвищити релевантність пошукових запитів.

2.1.2 Використання векторних представлень слів (Word2Vec, GloVe, BERT) для семантичного аналізу

Семантичний аналіз у прикладній лінгвістиці та інформатиці поєднує вивчення значення (семантики) з методами, які дозволяють реалізувати це на цифрових комп'ютерах. Семантика фокусується на аналізі сенсу тексту, тоді як обчислювальний аспект забезпечує ефективність і швидкість виконання цих процесів за допомогою сучасних алгоритмів та технологій.

Word2vec (рис.2.3) це техніка/модель для вбудовування слів для кращого представлення слів. Це метод обробки природної мови, який фіксує велику кількість точних синтаксичних і семантичних зв'язків слів. Це неглибока двошарова нейронна мережа, яка може виявляти слова-синоніми та пропонувати додаткові слова для часткових речень після навчання [19].

Неглибока нейронна мережа складається з одного прихованого шару між вхідним і вихідним рівнями, тоді як глибока нейронна мережа має кілька таких шарів. Вхідні дані проходять через вузли, а приховані й вихідні шари складаються з нейронів.

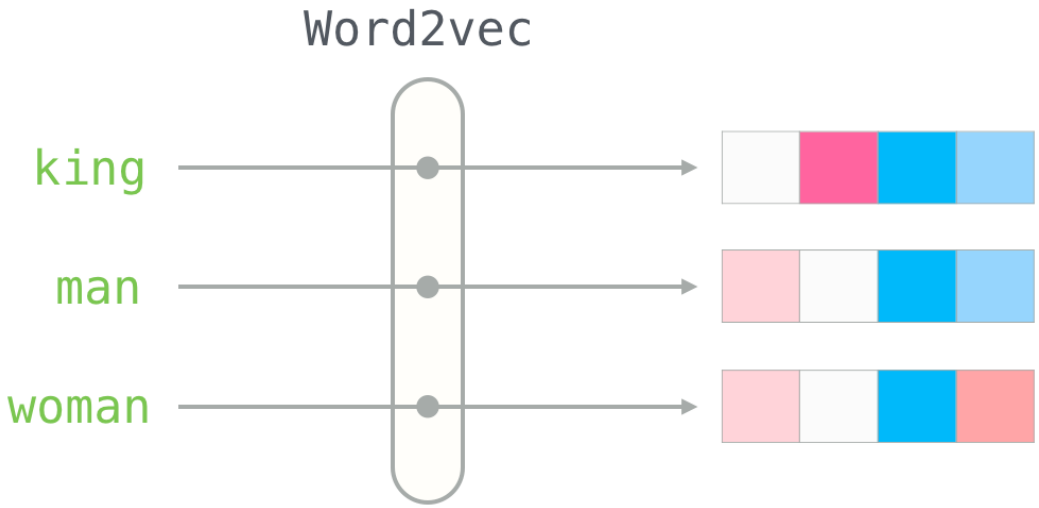

Рис. 2.3. Приклад роботи Word2vec

Word2vec є прикладом двошарової нейронної мережі: вона має вхідний шар,

один прихований шар і вихідний. Цю модель розробила команда під керівництвом Томаша Міколова в Google. Word2vec виявилася швидшою та ефективнішою, ніж метод прихованого семантичного аналізу, для представлення слів у числовій формі та роботи з контекстом.

GloVe та Word2Vec є схожими моделями представлення слів, які з'явилися майже одночасно й базуються на роботі з векторами слів.

Основна мета *GloVe* полягає у використанні статистики збігів ефективнішим чином. Модель зменшує різницю між твором векторів слів і логарифмом ймовірності їхньої спільної появи, застосовуючи стохастичний градієнтний спуск. Це дозволяє отримувати векторні уявлення, які відображають важливі лінійні залежності, наприклад, між супутниками планети чи поштовими кодами міст і їхніми назвами.

На відміну від GloVe, Word2Vec менше залежить від частоти спільної зустрічальності слів, використовуючи її лише для створення додаткових навчальних вибірок. GloVe, у свою чергу, враховує цю статистику, що дозволяє групувати вектори слів на основі їхньої глобальної схожості, а не лише контекстних залежностей. Таким чином, GloVe краще працює із загальною статистикою тексту, тоді як Word2Vec більше зосереджується на локальному контексті.

BERT, відомий як Bidirectional Encoder Representations from Transformers, є нейронною мережею, розробленою Google, яка продемонструвала високу ефективність у вирішенні різних завдань з обробки природної мови, включаючи відповіді на питання і машинний переклад. Код моделі є доступним для загального використання [20]. Структура системи продемонстрована в рис. 2.4.

Ця модель відрізняється двонаправленим підходом до контексту, що дозволяє їй краще розуміти залежності між словами у реченні. Завдяки цьому BERT став основою для багатьох сучасних рішень у сфері NLP, включаючи системи пошуку та діалогові агенти. Постійні вдосконалення та адаптація моделі забезпечують її ефективне застосування у різних сферах, таких як медична аналітика, автоматизована підтримка користувачів і контент-менеджмент.

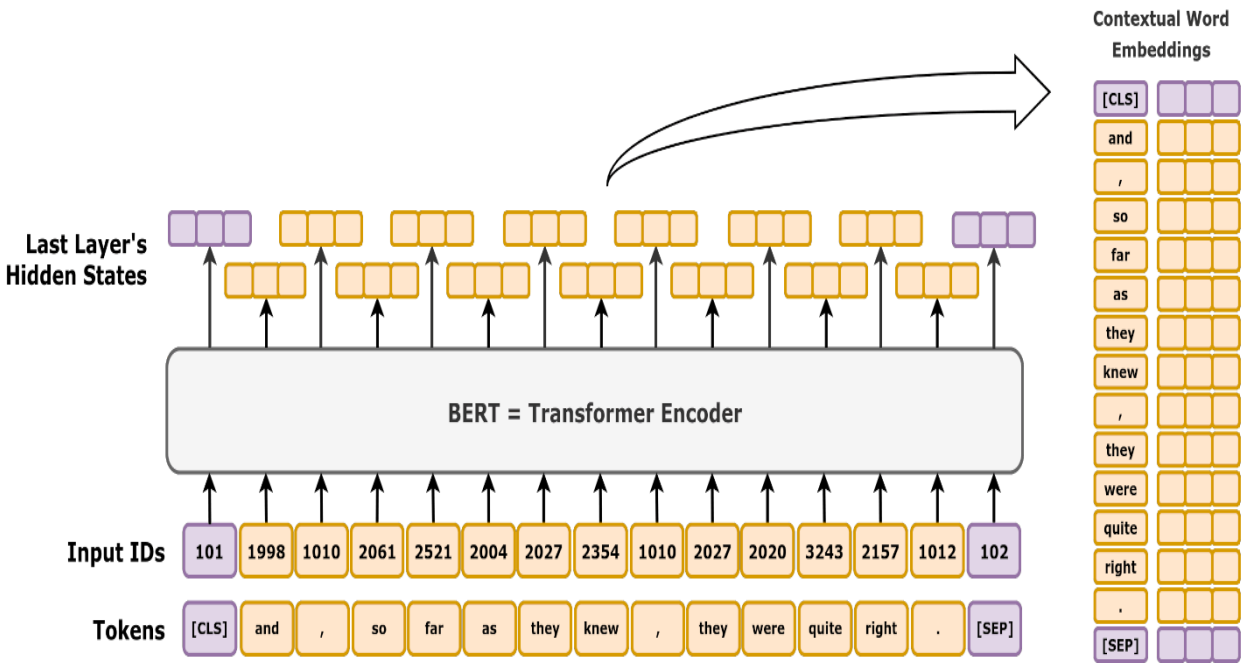

Рис. 2.4. Високорівнева структурна схема BERT.

BERT має свої обмеження через архітектуру, що базується лише на кодері без декодера. Це означає, що модель не здатна генерувати текст або робити підказки, оскільки двонаправлені моделі, до яких належить BERT, не можуть ефективно працювати без правого контексту. Генерація навіть коротких текстів із такими моделями потребує складних обчислень і значних ресурсів.

На відміну від традиційних нейронних мереж глибокого навчання, BERT вже пройшов попереднє навчання. Це дозволяє йому розуміти представлення слів, речень і основні семантичні зв'язки, що спрощує його адаптацію до конкретних завдань, наприклад класифікації настроїв. Для цього модель потрібно донавчити на менших наборах даних, адаптованих до конкретної сфери. Наприклад, для аналізу настроїв у фінансових текстах варто використовувати модель, спеціально попередньо навчену на фінансових даних.

Оригінальні ваги таких попередньо навчених моделей доступні для використання, наприклад, на платформах на зразок GitHub, що дозволяє дослідникам і розробникам інтегрувати їх у свої проєкти для вирішення різноманітних задач.

2.1.3 Аналіз емоцій (Sentiment Analysis) для кращого розуміння настрою користувача.

Аналіз настроїв — це процес висновків, вимірювання або розуміння образу вашого продукту, послуги чи бренду на ринку. Він аналізує людські емоції та почуття, інтерпретуючи нюанси у відгуках клієнтів, фінансових новинах, соціальних мережах тощо. Якщо це звучить надто складно, давайте вдосконалимо це далі [21].

Аналіз настроїв, також відомий як аналіз думок, фокусується на оцінці емоційного забарвлення тексту. З появою соціальних медіа люди почали активно ділитися своїм досвідом, враженнями та думками про продукти й послуги через різні онлайн-канали: блоги, соціальні мережі, коментарі, огляди та багато інших платформ. Цей процес залишає цифровий слід у вигляді текстових даних, які можуть містити позитивні, негативні або нейтральні відгуки.

Аналіз настроїв дозволяє зібрати ці дані й перетворити їх на цінну інформацію для бізнесу. Завдяки цьому бренди отримують змогу розуміти потреби та настрій своєї аудиторії, виявляти тенденції ринку й оцінювати свою позицію серед конкурентів. У підсумку, цей підхід допомагає зрозуміти, як споживачі сприймають бренд, продукт або послугу, що є важливим для прийняття стратегічних рішень.

2.2 Алгоритми інтеграції чат-ботів із внутрішніми системами компанії через API

2.2.1 Використання RESTful API для передачі даних між чат-ботом і корпоративними системами

RESTful API — це архітектура інтерфейсу прикладних програм (API), що використовує HTTP-запити для доступу до ресурсів та їхнього використання (Див рис. 2.5). Такими HTTP-запитами є *GET, PUT, POST і DELETE*. Вони дають змогу, відповідно, читати, змінювати, створювати й видаляти ресурси [22].

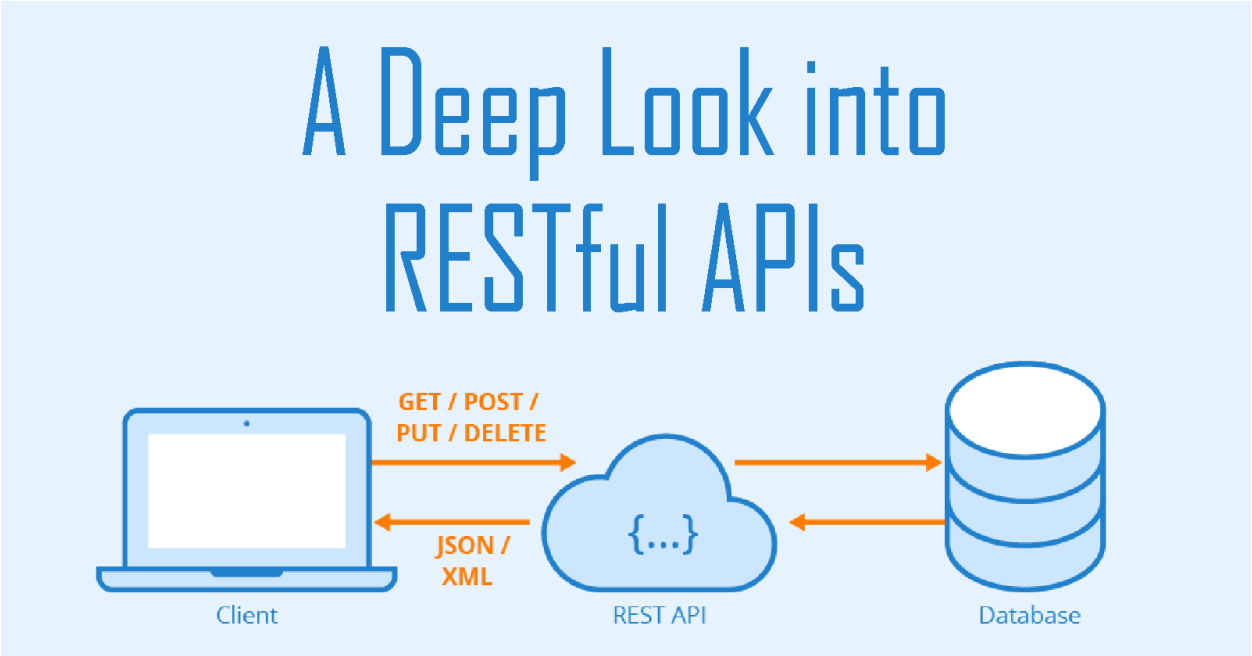

Рис. 2.5. *RESTful API*

RESTful API використовують для доступу до різних цифрових ресурсів, таких як дані, контент, алгоритми та мультимедійні файли, через URL-адреси в інтернеті. Це найпоширеніші API на сьогодні. У цьому матеріалі ми розглянемо, як вони працюють. REST є популярнішою технологією порівняно з іншими, оскільки потребує меншої пропускної здатності, що робить інтернет-з'єднання ефективнішим. RESTful API можна писати на різних мовах програмування, таких як JavaScript або Python.

Архітектура REST дозволяє створювати API, які дають змогу користувачам підключатися до хмарних сервісів, керувати ними та взаємодіяти з ними в розподіленому середовищі. RESTful API використовують такі компанії, як Amazon, Google, LinkedIn та X (Twitter).

Щоб API-сервіс був RESTful, потрібно дотримуватися шести обмежень:

1. *Уніфікований інтерфейс (UI):* Інтерфейс має бути єдиним, з унікальними ресурсами, які можна ідентифікувати за окремою URL-адресою.

2. *Клієнт-серверна архітектура*: Клієнтська частина надає інтерфейс користувача та збирає запити, а серверна забезпечує доступ до даних, керування

робочим навантаженням і безпеку. Вони можуть розвиватися незалежно одна від одної.

3. *Робота без збереження стану*: У запиті від клієнта до сервера має бути вся необхідна інформація для обробки запиту. Сервер не зберігає інформацію про стан клієнта.

4. *Кешування ресурсів*: Дані у відповіді на запит мають бути позначені як кешовані або некешовані.

5. *Багаторівнева система:* REST допускає архітектуру з ієрархічних рівнів, де кожен компонент взаємодіє лише з безпосередньо пов'язаним рівнем.

6. *Код на вимогу*: API-інтерфейси REST можуть завантажувати та виконувати код у формі аплетів або сценаріїв, розширюючи функціональність клієнтської частини.

RESTful API розбиває транзакцію на серію невеликих модулів, кожен з яких виконує певну частину транзакції. Це надає розробникам гнучкість, але може бути складно розробити RESTful API з нуля. Тому деякі компанії надають свої моделі, наприклад, Amazon S3, Cloud Data Management Interface (CDMI) та OpenStack Swift.

RESTful API використовує команди для отримання ресурсів. Стан ресурсу в будь-який момент часу називається представленням ресурсу. RESTful API використовує методології HTTP, визначені протоколом RFC 2616.

Застосування

Хмарні програми: RESTful API корисні у хмарних програмах, оскільки їхні виклики не зберігають стан. Це дозволяє легко перерозподіляти та масштабувати компоненти відповідно до змін навантаження.

Хмарні сервіси: REST корисний у хмарних сервісах, оскільки дозволяє контролювати, як декодується URL-адреса для прив'язки до сервісу через API. Хмарні обчислення та мікросервіси роблять розробку RESTful API стандартом у майбутньому.

Використання в інтернеті: REST не прив'язано до клієнтської технології, тому ці API можуть бути доступні з вебпроєктів, програм iOS, IoT або Windows Phone.

Ви можете побудувати інфраструктуру для своєї організації без прив’язки до певного стека на боці клієнта.

2.2.2 Методи автентифікації та авторизації для захисту доступу до систем (OAuth, JWT)

Аутентифікація - це процес підтвердження ідентичності користувача. Це визначає, хто саме намагається отримати доступ до системи чи додатку. Зазвичай процедура аутентифікації включає в себе введення ім'я користувача та пароля. Однак вони можуть бути більш складні, наприклад, біометричні дані (відбиток пальця, розпізнавання обличчя) чи використання двофакторної аутентифікації (пароль та код, отриманий на мобільний телефон) [23].

Аутентифікація є критичним елементом безпеки для будь-якої системи, оскільки вона забезпечує, що лише авторизовані користувачі можуть отримати доступ до своїх облікових записів або особистих даних. Вона зазвичай включає перевірку ідентичності через логін та пароль, але також може включати додаткові методи, такі як двофакторна аутентифікація (2FA), біометричні дані або одноразові паролі.

Правильно налаштована аутентифікація запобігає несанкціонованому доступу, захищає конфіденційність даних та мінімізує ризики зловмисних атак, таких як крадіжка облікових записів або доступ до чутливих даних. Важливо, щоб системи аутентифікації регулярно оновлювались, а користувачі застосовували сильні паролі і додаткові методи захисту для максимального рівня безпеки.

Авторизація - це процес визначення, до яких ресурсів або функціональності має доступ користувач після успішної аутентифікації. Після визначення ідентичності користувача система вирішує, які дії він може виконувати та які ресурси він може переглядати чи редагувати. Наприклад, адміністратор системи має більше прав доступу, ніж звичайний користувач [23].

Авторизація — це процес, що визначає рівень доступу користувача до ресурсів або функцій системи після того, як його ідентичність була підтверджена через

аутентифікацію. Це забезпечує гарантію, що користувач має право лише на ті ресурси та дії, які йому дозволено виконувати.

Наприклад, в системах, де є кілька рівнів доступу (адміністратори, користувачі, гості), авторизація контролює, хто може редагувати, переглядати або видаляти інформацію, а хто може лише читати або отримувати базовий доступ. Авторизація, як правило, здійснюється на основі ролей або правил, заданих для кожного користувача чи групи.

Цей механізм часто поєднується з аутентифікацією для створення більшої безпеки в системах. Якщо аутентифікація підтверджує, хто користувач, то авторизація визначає, що він може робити в межах доступних ресурсів.

У відповідь на зростаючу кількість онлайн-сервісів та додатків компанії удосконалили свої системи аутентифікації та авторизації. Двофакторна аутентифікація, використання токенів і інші сучасні технології допомагають підвищити рівень безпеки. Проте, важливо також інформувати користувачів про те, як захистити свої акаунти, створюючи складні паролі та обираючи надійні методи аутентифікації.

OAuth, або Open Authorization – це протокол авторизації, який надає одним веб-додаткам можливість отримати доступ до ресурсів іншого додатка від імені користувача, без необхідності розкривати свої облікові дані [24].

OAuth – забезпечити безпеку і зручність для користувачів і додатків. Замість того щоб ділитися своїми обліковими даними з кожним сервісом окремо, користувачі можуть дозволити доступ до своїх даних тільки певним частинам інформації. Застосунки отримують доступ тільки до тієї інформації, яка їм дійсно потрібна для роботи див рис 2.6.

Цей протокол аутентифікації дозволяє користувачам контролювати, які дані передаються стороннім додаткам, мінімізуючи ризики витоку конфіденційної інформації. OAuth активно використовується популярними платформами, такими як Google, Facebook та Microsoft, для спрощення процесу входу до облікових записів.

Завдяки своїй гнучкості та надійності, він став стандартом у багатьох веб- та мобільних додатках для забезпечення безпечного доступу до ресурсів.

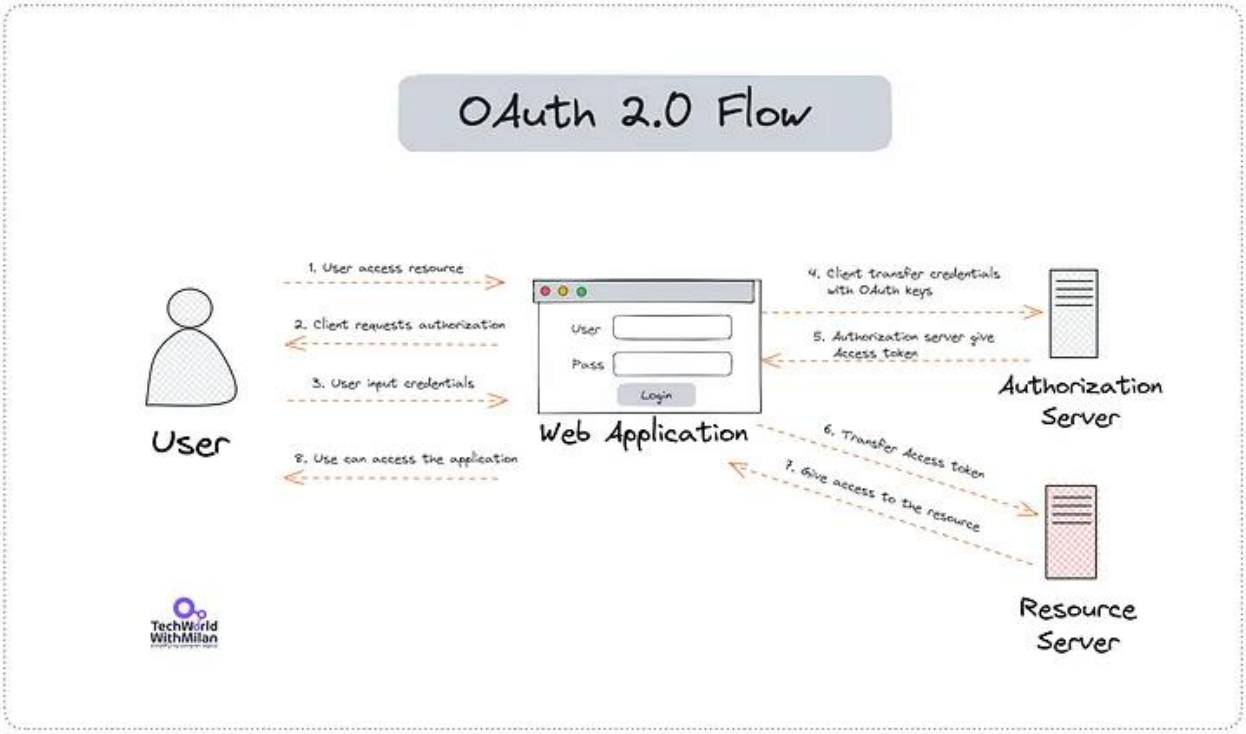

Рис. 2.6. Приклад роботи OAuth [26]

В основі роботи OAuth лежить ідея делегування доступу. Замість того щоб давати свій пароль, ви просто дозволяєте додатку використовувати певні ресурси. Для цього ви надаєте дозвіл на доступ, і додаток отримує спеціальний ключ, або токен, який підтверджує ваш дозвіл.

Ви можете увійти на веб-сайт або в додаток через свій акаунт на іншому веб-сервісі, такому як Google або Facebook. Замість того щоб вводити свої облікові дані, ви підтверджуєте свою особистість на зовнішньому сайті, і цей сайт створює спеціальний токен, який передають веб-сайту або застосунку, до якого ви хочете отримати доступ.

JsonWebToken (JWT) — це компактний, безпечний для URL-адрес, відкритий стандарт, який визначає метод безпечної передачі інформації шляхом кодування та підпису даних JSON. JWT використовуються для передачі різних типів інформації,

але в основному його використовують для доставки «вимог», які є фрагментами інформації. На практиці вимоги, швидше за все, міститимуть корисну інформацію про користувача, що найчастіше використовується для авторизації та автентифікації [27]. Приклад на Рис 2.7.

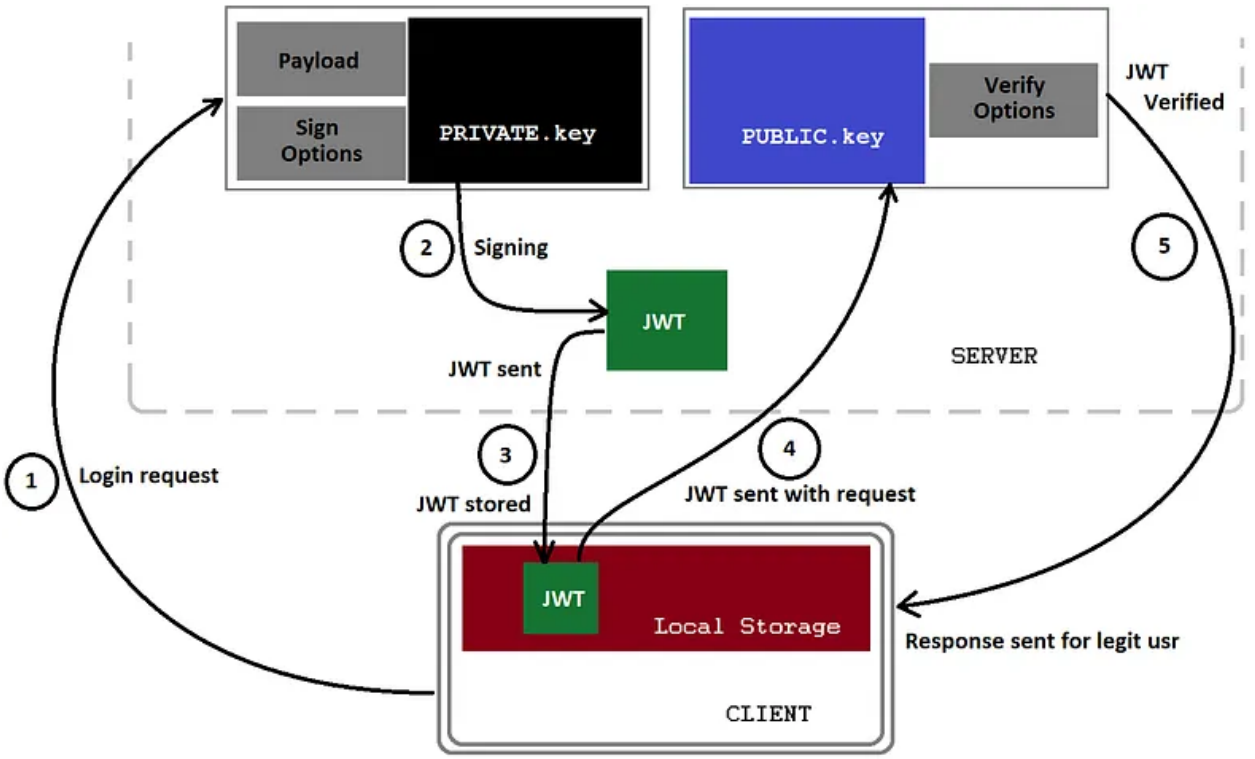

Рис. 2.7. Схема роботи JsonWebToken [28].

Процес автентифікації з використанням JWT (JSON Web Token) можна пояснити таким чином:

1. *Користувач надає облікові дані.* Перший крок – це введення облікових даних користувачем, наприклад, логіна і пароля.

2. *Перевірка даних на сервері.* Сервер перевіряє ці облікові дані проти бази даних, щоб підтвердити їх правильність.

3. *Генерація JWT*. Якщо дані правильні, сервер створює JWT, що містить вимоги та права користувача, і підписує його за допомогою секретного ключа.

4. *Зберігання JWT*. Токен передається користувачу, який зберігає його у браузері (як cookie або в локальному сховищі).

5. *Запит до захищеного ресурсу*. Коли користувач звертається до захищеного ресурсу, його запит містить JWT. Запит надіслається серверу, щоб отримати доступ до ресурсу.

6. *Перевірка JWT.* Сервер перевіряє підпис токену, щоб впевнитися, що він не був змінений, і що користувач має права доступу.

7. *Надання доступу*. Якщо перевірка пройдена успішно, користувач отримує доступ до ресурса. Якщо перевірка не пройдена, сервер відмовляє в доступі (помилка 401), або якщо токен прострочений, користувач повинен авторизуватись знову.

Вразливості JWT: Атаки на JWT часто спричинені помилками у реалізації, зокрема, якщо секретний ключ не захищений належним чином. Зловмисники можуть отримати доступ до ключа і виконати атаку, що дозволить обійти автентифікацію та авторизацію, що призведе до викрадення або зміни даних.

2.2.3 Оптимізація швидкості обробки запитів через паралельну обробку та кешування.

Сучасні інформаційні системи (ІС) для швидкісної обробки баз даних, завдяки доступу до інтернету, активно використовують комп’ютери з паралельною обробкою даних.

Паралельна обробка запитів (Parallel Data Query, PDQ) - це технологія, яка дозволяє розподілити обробку одного складного запиту на кілька процесорів , мобілізувати для його виконання максимально доступні системні ресурси, зменшити час отримання результату [29].

Технологія PDQ (Parallel Data Query) дозволяє значно зменшити час обробки даних та забезпечити оперативну реакцію на термінові запити. Вона знижує кількість проблем, пов’язаних з обробкою великих таблиць, завдяки використанню фрагментації, паралельної обробки та виконання адміністративних дій в реальному часі. Це дозволяє підвищити ефективність роботи з великими базами даних, зокрема

на багатопроцесорних платформах, де час виконання запитів може скорочуватися в десятки разів.

На однопроцесорних системах та нефрагментованих таблицях виграш у продуктивності досягається завдяки паралельному доступу до дисків та оптимальному використанню пам'яті. Застосування PDQ також розширює коло можливих додатків, оскільки дозволяє працювати з великими даними та швидко їх обробляти.

Сучасний підхід до "швидкісної обробки баз даних" включає використання новітніх засобів збору та обробки інформації, що підвищує швидкість перетворення даних та отримання результатів. Для розробки програмного забезпечення, яке підтримує PDQ, рекомендується використовувати мови програмування, які підтримують паралельну обробку даних, а також СУБД, веб-сервери та фреймворки, що забезпечують паралельний доступ до баз даних.

Кешування – це механізм зберігання часто використовуваних даних або файлів у тимчасовому сховищі, яке забезпечує швидший доступ у майбутньому. У контексті мережі кешування дозволяє зберігати дані, такі як зображення, стилі CSS або сценарії JavaScript, на пристрої користувача або проміжному сервері. Це дозволяє повторно використовувати ці дані, коли вони знову запитуються, не потребуючи повторного доступу до вихідного джерела [30].

Переваги системи кешування

Впровадження системи кешування в мережах дає численні переваги:

1. *Покращена продуктивність*: Зменшується час завантаження веб-сайтів, оскільки дані не потрібно завантажувати з віддаленого сервера.

2. *Підвищена доступність*: у разі збоїв або нестабільного з’єднання дані з кешу можна отримати локально, що забезпечує безперебійний доступ.

3. *Зменшення навантаження на сервер*: Кешування знижує кількість запитів до вихідного сервера, підвищуючи загальну продуктивність.

4. *Економія пропускної здатності*: Використання кешованих даних знижує обсяг переданих через мережу даних, що зменшує витрати на пропускну здатність.

5. *Покращений досвід користувача*: Швидший доступ до контента покращує взаємодію з веб-сайтом.

Типи систем кешування

Існує два основних типи кешування:

- *Кеш на стороні браузера*: Зберігає дані безпосередньо в браузері користувача, що дозволяє повторно використовувати їх без доступа до сервера.
- *Кеш на стороні сервера*: Дані зберігаються на проміжному сервері, що дозволяє швидше надавати часто запитуваний контент.

Фактори, що впливають на ефективність кешування

Ефективність кешування залежить від кількох факторів:

- *Політика кешування*: Визначає, як довго зберігаються дані в кеші.
- *Структура кешу*: Вибір правильних структур даних для швидкого доступу.
- *Алгоритм заміщення*: Визначає, які дані повинні бути замінені, коли кеш заповнений.
- *Конфігурація обладнання*: Впливає на швидкість та ємність кешуючих серверів.

Оптимізація кешування для максимального ефекту

Для максимального покращення продуктивності необхідно:

- Налаштувати політику кешування для різних типів даних.
- Використовувати ефективні структури даних для швидкого доступу.
- Вибирати оптимальні алгоритми заміщення, як-от LRU або LFU.
- Інвестувати в потужне обладнання для кешуючих серверів.

Система кешування є важливим компонентом сучасної веб-архітектури, який значно покращує продуктивність, доступність та досвід користувача. Впровадження та оптимізація кешування дозволяють веб-сайтам завантажуватися швидше, бути більш надійними та надавати користувачам плавне та приємне враження від перегляду. Розуміння принципів кешування та налаштування системи відповідно до

потреб конкретного веб-сайту забезпечують максимальну ефективність та задоволеність користувачів [30].

2.2.4 Інтеграція з базами даних (SQL, NoSQL) для швидкого доступу до даних.

Бази даних є основою для зберігання та організації даних у веб-додатках. Існують два основних типи баз даних: *SQL* (структуровані) та *NoSQL* (неструктуровані). Обидва типи мають свої переваги та особливості використання залежно від конкретних вимог додатку.

SQL бази даних — це реляційні бази даних, які використовують мову структурованих запитів (Structured Query Language, SQL) для створення, керування та обробки даних. Вони складаються з таблиць, де дані представлені у вигляді рядків (записів) та стовпців (атрибутів) [31].

SQL є потужним інструментом для роботи з даними, який дозволяє виконувати широкий спектр операцій, включаючи вибірку, вставку, оновлення та видалення. Окрім цих основних функцій, SQL також підтримує більш складні операції, такі як злиття таблиць (JOIN), групування даних (GROUP BY), фільтрація (WHERE) та сортування (ORDER BY), що дозволяє здійснювати гнучкий аналіз даних. До найбільш популярних SQL баз даних відносяться MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database та SQLite.

NoSQL бази даних — це нереляційні системи зберігання даних, які не використовують стандартний SQL для обробки даних [31].

NoSQL бази даних відрізняються високою гнучкістю, швидкістю та масштабованістю, що дозволяє їм ефективно працювати з великими наборами даних, включаючи неструктуровані або напівструктуровані дані. Це робить їх ідеальними для таких застосувань, де дані змінюються або мають різні формати, як-от в аналізі соціальних медіа, великих даних або обробці інформації з Інтернету речей. Вони також забезпечують горизонтальне масштабування, що дозволяє обробляти збільшення навантаження без значних змін у архітектурі системи.

При виборі між SQL і NoSQL базами даних важливо враховувати тип даних і вимоги до консистентності та масштабованості. SQL бази даних, як-от MySQL та PostgreSQL, ідеальні для роботи з добре структурованими даними, де важлива сильна консистентність і підтримка складних транзакцій. В той час як NoSQL, такі як MongoDB або Cassandra, зазвичай використовуються для роботи з даними, які не потребують жорсткої структури, а також для зберігання та обробки великих обсягів даних, що потребують швидкої горизонтальної масштабованості. Короткий приклад структури на рис 2.8.

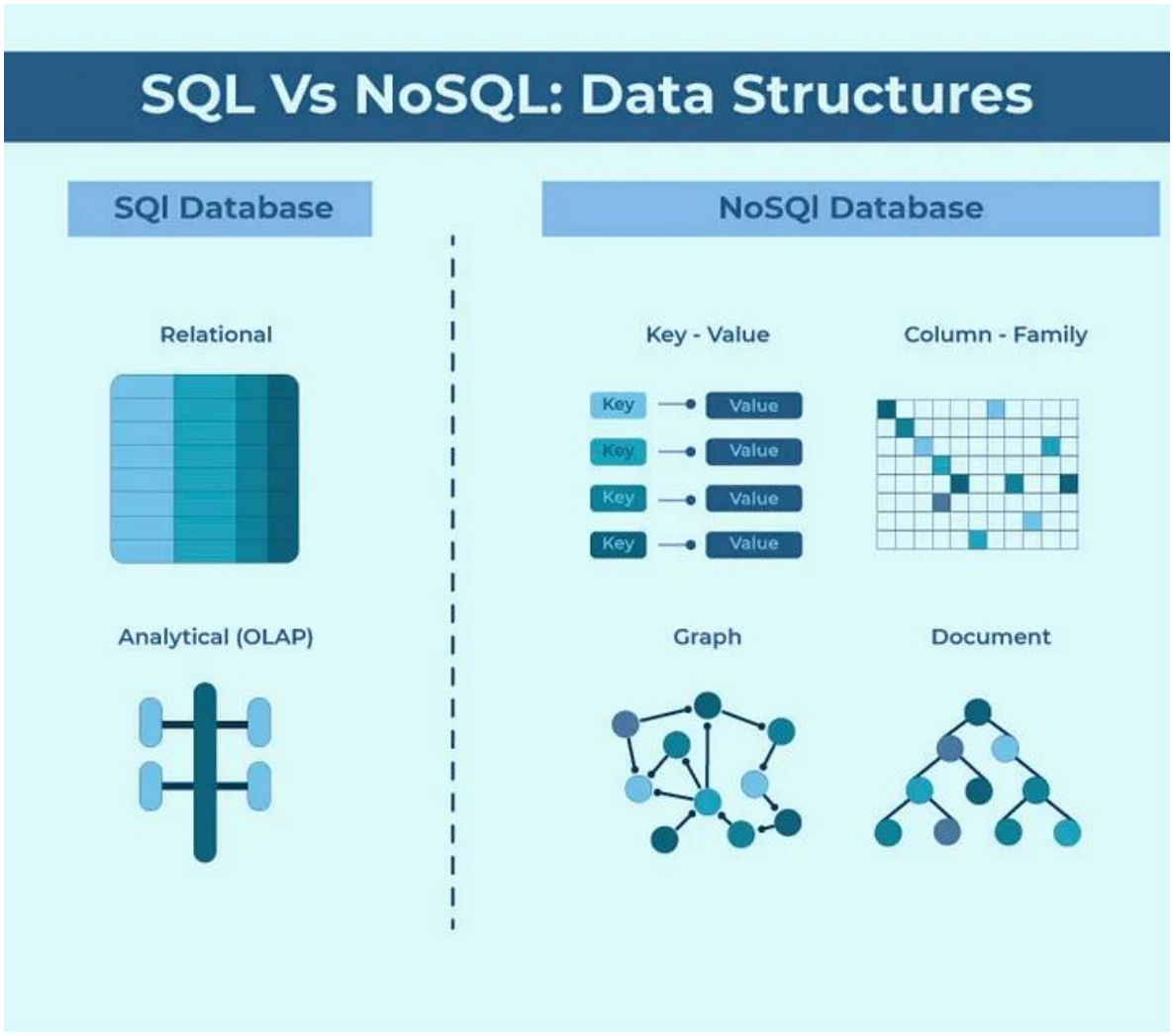

Рис. 2.8. Структури SQL та NoSQL [32].

2.2.5 Взаємодія з сторонніми сервісами (платіжні системи, поштові сервіси тощо) для розширення функціональних можливостей чат-ботів.

Чат-боти перетворюються на важливий інструмент для бізнесу, адже вони значно покращують взаємодію з клієнтами. Вони дають змогу компаніям оперативно відповідати на запити користувачів і забезпечують безперервну підтримку, що важливо в умовах конкурентного середовища. Чат-боти допомагають підтримувати взаємодію 24/7, розширювати охоплення аудиторії та створювати довгострокові стосунки з клієнтами, що може безпосередньо вплинути на успішність продажів та розвиток бізнесу. Це стає особливо важливо для компаній, які прагнуть зберегти конкурентні переваги у швидко змінному цифровому середовищі.

Чат-ботами називають спеціальні програми, які взаємодіють із користувачем, імітуючи спілкування з живою людиною. Вони можуть збирати дані, надавати інформацію або відповідати на поширені запитання, а в нестандартних випадках підключати живих менеджерів [33].

Чат-боти мають можливість надсилати повідомлення в месенджери, стилізуючи їх під певний тонус спілкування бренду чи експерта. Це дозволяє створити персоналізовану взаємодію з користувачем, що сприяє кращому сприйняттю повідомлень. Для цього використовуються смайли, спеціальні кнопки для взаємодії та налаштовані сценарії. Ланцюжки повідомлень продумані та організовані логічно, що дозволяє автоматично вести користувача через етапи комунікації, сприяючи ефективному вирішенню запитів або продажам. Всі ці елементи запрограмовані заздалегідь, що робить взаємодію з ботом ефективною і легко керованою.

Насамперед усе розмаїття розумних помічників можна розділити на запрограмованих і таких, що самонавчаються. У першому випадку створення такого чат-бота доступне навіть малому бізнесу та окремим експертам. Для цього існують сервіси-конструктори, що дають змогу зібрати простого бота безкоштовно за 1 день. Його недоліком буде негнучкість і невисока схожість на живу людину [33].

Переваги чат-ботів для бізнесу: ефективність і зручність

У сучасному бізнес-середовищі, де конкуренція та вимоги клієнтів постійно зростають, автоматизація процесів стає надзвичайно важливою. Чат-боти є незамінними помічниками, які можуть виконувати безліч завдань, підвищуючи ефективність бізнесу та знижуючи витрати. Ось кілька способів, як чат-боти можуть бути корисними для вашого бізнесу:

1. *Автоматизація обробки запитів*: Зі зростанням бізнесу збільшується потік запитів від клієнтів, що може бути надмірним для традиційних методів обробки. Чат-боти дозволяють автоматизувати цей процес, швидко реагуючи на запити без участі людини, що значно підвищує продуктивність.

2. *Навчання та розвиток співробітників*: Чат-бот може стати надійним помічником у навчанні працівників, надаючи доступ до бази знань цілодобово. Він може нагадувати про нові курси, надсилати завдання та тестування без необхідності людського втручання, що дозволяє співробітникам навчатися у будь-який зручний час.

3. *Бот-візитка компанії*: Чат-бот може бути ефективною візиткою компанії, яка знайомить користувачів із брендом, продукцією, контактною інформацією та режимом роботи. Це не лише полегшує взаємодію, а й зберігає контакт з потенційними клієнтами, оперативно відповідаючи на поширені запитання.

4. *Технічна підтримка*: Використання чат-ботів для технічної підтримки значно скорочує витрати на розширення штатів підтримки. Бот здатний автоматично відповідати на часто задавані питання та оперативно працювати з великими обсягами даних, забезпечуючи безперервну допомогу 24/7, навіть у нічний час.

5. *Прогрів лідів та супровід продажів*: Чат-боти, налаштовані для ведення воронки продажів, допомагають не лише знайомити потенційних клієнтів з продуктом, а й проводити їх від початкового інтересу до завершення покупки. Вони можуть також збирати зворотний зв’язок та надсилати персоналізовані пропозиції, що підвищує ймовірність успішної угоди.

6. *Швидка реакція на запити*: Миттєва реакція на звернення користувачів значно покращує досвід взаємодії, роблячи спілкування більш природним і довірчим.

Чат-боти, виконуючи свою роль у месенджерах, здатні налагодити більш особисті та неформальні зв’язки з клієнтами.

7. *Розширення охоплення і утримання аудиторії***:** Завдяки популярності месенджерів, чат-боти стають потужним інструментом для підтримки і розширення аудиторії. Вони допомагають утримувати клієнтів у воронці продажів і забезпечують постійну взаємодію з брендом.

Таким чином, чат-боти не лише полегшують повсякденні бізнес-операції, але й покращують досвід клієнтів, дозволяючи компаніям працювати ефективніше.

2.3 Алгоритми самонавчання чат-бота

Навчання з підкріпленням (RL) — це підхід до машинного навчання, який дозволяє агенту вивчати оптимальну поведінку через взаємодію з навколишнім середовищем. Окрім того що підхід дозволяє уникнути проблем з попереднього пункту, він також передбачає навчання методом проб і помилок, отримання зворотного зв’язку у вигляді винагород або штрафів і оптимізацію дій агента для максимізації сукупних винагород [34].

Що стосується чат-ботів, методи підкріплювального навчання (RL) пропонують кілька переваг і вирішують певні проблеми:

1. Адаптивна та персоналізована взаємодія: RL дозволяє чат-ботам адаптуватися до різних уподобань користувачів і оптимізувати їхні відповіді на основі відгуків у реальному часі. Постійно вивчаючи та оновлюючи свої політики, чат-боти на основі RL можуть забезпечувати більш персоналізовані та контекстуально відповідні розмови.

2. Дослідження та використання: методи RL дозволяють чат-ботам знаходити баланс між пошуком нових відповідей і використанням відомих ефективних стратегій. Це дозволяє чат-боту виявляти нові моделі розмов і з часом покращувати свою продуктивність.

Робота з недетермінованим середовищем: розмовні взаємодії можуть бути дуже непередбачуваними, що ускладнює створення систем на основі правил. Чат-боти на основі RL можуть працювати з недетермінованим середовищем, навчаючись на досвіді та відповідним чином адаптуючи свої відповіді [34].

Навчання на відгуках людей: Підкріплювальне навчання може використовувати відгуки людей для покращення продуктивності чат-бота. За допомогою методів, таких як навчання з підкріпленням за участю людини або навчання на демонстраціях, чат-боти можуть навчатися на відгуках експертів або користувачів, покращуючи свої комунікативні здібності.

Динамічні розмови, що розвиваються: Розмови є динамічними та можуть змінюватися з часом. Підкріплювальне навчання дозволяє чат-ботам адаптуватися та вчитися на змінних контекстах, що дозволяє їм підтримувати послідовні та релевантні розмови протягом кількох ходів.

Незважаючи на ці переваги, є деякі проблеми, пов’язані з чат-ботами на основі RL, пов’язані з складністю вибірки: навчання з підкріпленням часто вимагає великої кількості взаємодій із середовищем для вивчення ефективної політики. У випадку чат-ботів це може призвести до тривалих розмов з користувачами, що не завжди можливо чи ефективно [34].

Ще однією складністю є визначення правильного балансу між дослідженням нових стратегій розмови та використанням відомих ефективних відповідей. Це нетривіальне завдання, оскільки чат-боту потрібно знайти баланс, щоб не застрягти в неоптимальних шаблонах розмови чи повторенні тих самих фраз.

Розробка відповідних функцій винагороди для чат-ботів також може бути складною. Сигнал винагороди повинен відображати якість розмови, яка за своєю суттю є суб’єктивною та залежить від контексту. Побудова функцій винагороди, які б узгоджувалися з уподобаннями людини, є постійною проблемою дослідження.

Етичні міркування: Чат-боти, засновані на підкріплювальному навчанні (RL), можуть навчатися на упереджених або шкідливих даних, що призводить до непередбачуваних наслідків. Забезпечення етичної поведінки та усунення

упереджень у процесі навчання має вирішальне значення, щоб запобігти створенню чат-ботом неналежних або шкідливих відповідей.

Підсумовуючи, проблема створення чат-бота на основі методів підкріплювального навчання є актуальною через його потенціал для створення адаптивних, персоналізованих та контекстно-релевантних розмовних агентів. Хоча існують труднощі, які потрібно подолати, застосування методів RL до чат-ботів є перспективним для просування в галузі розуміння природної мови та людської взаємодії.

2.4 Висновки до розділу

У цьому розділі наведено огляд сучасних технологій для інтеграції та алгоритмізації чат-ботів з акцентом на внутрішню автоматизацію процесів, що дозволяє значно підвищити ефективність взаємодії з користувачами та бізнес-процесами. Спочатку було розглянуто основні методи обробки природної мови (NLP), такі як стемінг, лематизація, токенізація та видалення стоп-слів, які є основними інструментами для попередньої обробки текстових даних. Вивчено функції цих методів як у створенні моделей, що допомагають в інтерпретації тексту, так і в підготовці текстового матеріалу для аналізу. Такі методи дають змогу спростити обробку мовних одиниць, знижуючи складність і покращуючи точність подальших етапів обробки. Особливу увагу приділено використанню векторних моделей представлення слів (Word2Vec, GloVe та BERT), які забезпечують більш точне уявлення про значення слів у контексті, що важливо для аналізу настрою, семантичного аналізу та контекстуальних взаємодій. Ці моделі дозволяють чат-ботам не лише розпізнавати слова, але й правильно інтерпретувати їх у різних ситуаціях.

У наступному розділі розглядаються алгоритми використання RESTful API для інтеграції чат-ботів з бізнес-системами, що є важливим етапом у забезпеченні їхньої функціональності в рамках компанії. Чат-боти можуть отримувати і

передавати дані через API, що дозволяє взаємодіяти з іншими внутрішніми і зовнішніми системами, забезпечуючи автоматизацію бізнес-процесів. Окремо розглядаються методи авторизації та аутентифікації, які захищають доступ до внутрішніх систем і забезпечують безпеку взаємодії. Використання таких технологій, як JWT та OAuth, дозволяє ефективно управляти доступом до даних і знижує ризики витоку інформації. Обговорюються також методи збільшення швидкості обробки запитів, що включають кешування і паралельну обробку запитів, що значно зменшує навантаження на систему та прискорює взаємодію з користувачем.

Також розглядається, як чат-боти можуть взаємодіяти зі сторонніми сервісами, такими як платіжні системи, поштові сервіси, системи для обробки запитів або сторонні бази даних, для розширення їх функціональності. Це дає можливість чат-ботам не лише обробляти запити, але й виконувати транзакції, надавати послуги, а також інтегруватися з базами даних (як SQL, так і NoSQL) для забезпечення швидкого доступу до даних, що дозволяє здійснювати операції у реальному часі.

Вивчення алгоритмів самонавчання чат-ботів, зокрема методів навчання з підкріпленням (RL), стало важливим етапом дослідження. Ці методи дозволяють чат-ботам вдосконалювати свої відповіді, враховуючи відгуки користувачів. Завдяки навчанню з підкріпленням чат-боти можуть адаптувати свої стратегії взаємодії, покращуючи точність і релевантність відповідей. Ці алгоритми дозволяють чат-ботам поступово підлаштовуватися під уподобання користувачів, що підвищує ефективність їх роботи та дозволяє більш точно відповідати на складні запити.

У результаті в цьому розділі було систематизовано, досліджено та рекомендовано стратегії підвищення ефективності чат-ботів за допомогою сучасних методів і технологій. Підвищення ефективності чат-ботів дозволяє знизити витрати на обслуговування клієнтів, автоматизувати значну частину процесів, що в кінцевому підсумку знижує навантаження на компанії та підвищує якість обслуговування.

РОЗДІЛ 3
РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНА ІНТЕНГРАЦІЯ ЧАТ-БОТА ДЛЯ АВТОМАТИЗАЦІЇ ВНУТРІШНІХ ПРОЦЕСІВ У КОМПАНІЯХ

3.1 Визначення функціональних вимог

3.*1.1 Аналіз потреб користувачів (студентів і викладачів).*

Метою автоматизації внутрішніх бізнес-операцій є підвищення продуктивності, зменшення навантаження на працівників та забезпечення швидкого доступу до необхідної інформації. Обробка стандартних запитів співробітників, отримання доступу до поточних даних та комунікація з багатьма підрозділами - все це труднощі, з якими стикаються сучасні компанії.

У багатьох бізнес-середовищах такі завдання, як управління документами, пошук інформації про дати зустрічей та отримання доступу до навчальних матеріалів, є рутинними. Розглянемо приклад університету, де чат-бот може бути використаний для полегшення комунікації між студентами, викладачами та адміністративними системами, щоб краще розуміти їхні потреби.

*Потреби студентів***:**

1. *Швидкий доступ до розкладу занять*.

Студенти іноді опиняються в ситуаціях, коли їм потрібно негайно дізнатися про будь-які зміни в розкладі занять. Це можна порівняти з тим, що потрібно працівникам бізнесу, щоб мати доступ до календаря конференцій чи зустрічей. *Приклад потреби*: "Які заняття у групи ІПм-23-1 на понеділок?"

2. Доступ до навчальних матеріалів.

Студентам потрібно швидко знаходити лекційні матеріали чи завдання. У компаніях це може бути доступ до інструкцій, корпоративних політик або ресурсів для навчання персоналу.

Приклад потреби: "Де знайти лекцію з основ програмування?"

3. *Контактна інформація адміністрації***.**

Звернення до адміністративного персоналу чи пошук відповідального за певні завдання є звичною потребою як для студентів, так і для співробітників компаній. *Приклад потреби*: "Хто займається оформленням довідок для відряджень?"

Потреби викладачів:

1. *Актуальний розклад***.**

Викладачам важливо мати доступ до точного розкладу, що включає інформацію про аудиторії, час занять і групи. У компаніях це аналогічно потребі керівників у доступі до планів зустрічей або управлінських завдань.

2. *Доступ до списків студентів.*

Викладачам потрібен список студентів, які відвідують їхні курси, щоб організовувати роботу груп. У корпоративному середовищі це може бути доступ до списків учасників проєкту чи співробітників у відділі. *Приклад потреби*: "Хто записаний на мій курс із програмування?"

3. *Обмін навчальними матеріалами***.**

Подібно до співробітників компаній, які обмінюються презентаціями чи файлами, викладачі потребують платформи для зручного розповсюдження лекцій, завдань і додаткових ресурсів.

Приклад потреби: "Як поділитися презентацією з моїми студентами?"

3.1.2 Визначення основних функцій чат-бота.

На основі аналізу потреб користувачів ми визначили основні функції чат-бота, які забезпечать ефективну автоматизацію внутрішніх процесів. Задоволення потреб користувачів, зв'язок з бізнес-процесами та забезпечення зручності використання - ось цілі цих функцій.

1. *Пошук розкладу занять чи заходів***.**

Одна з основних функцій чат-бота – забезпечення швидкого доступу до розкладу занять, зустрічей або інших заходів. Чат-бот повинен обробляти запити, сформульовані природною мовою, наприклад:

- "Які пари у групи ІП-24-3 сьогодні?"

- "Який у мене розклад на середу?"

Відповіді формуються на основі попередньо завантажених даних з індексів, що дозволяє оперативно знаходити потрібну інформацію.

2. *Надання доступу до навчальних або корпоративних матеріалів***.**

Користувачі повинні мати можливість отримувати необхідні файли (лекції, презентації, методичні рекомендації) за запитом. Для цього реалізується функціонал пошуку в базі даних, де кожен документ має чітку класифікацію:

- "Дай лекцію з програмування."
- "Знайди презентацію з маркетингу."

Чат-бот надає посилання на файл або короткий витяг із нього.

3. *Пошук контактної інформації.*

Ще однією важливою особливістю є надання споживачам миттєвого доступу до контактів адміністрації, інструкторів або інших відповідальних осіб. Це зменшує навантаження на адміністративний персонал та економить час.

Приклади запитань:

- "Хто відповідальний за оформлення довідок?"
- "Як зв’язатися з деканатом інституту ІІТ?

4. *Підтримка багатомовності.*

Для університетів із міжнародними студентами або компаній із глобальною присутністю важливо реалізувати підтримку багатьох мов. Чат-бот повинен відповідати користувачам тією мовою, якою був сформульований запит.

5. *Інтеграція з внутрішніми системами***.**

Для того, щоб чат-бот працював, він має бути інтегрований з корпоративними базами даних, CRM або системами управління навчанням (LMS). Це дозволить вам пропонувати як загальну інформацію, так і індивідуальні відповіді, наприклад:

- "Які завдання мені потрібно виконати для курсу 'Алгоритми'?"
- "Які документи залишилося здати для завершення реєстрації?"

6. *Забезпечення безпеки даних.*

Для гарантування конфіденційності та захисту персональних даних чат-бот

повинен працювати із зашифрованими базами даних, використовувати захищені протоколи для передачі інформації (наприклад, HTTPS) та автентифікацію користувачів.

3.1.3 Формування структури даних для зберігання та пошуку.

Створення прозорої структури зберігання даних, яка дозволяє швидко знаходити та обробляти запити, має вирішальне значення для належного функціонування чат-бота. Основними компонентами структури є:

1. *Категоризація даних*

Дані мають бути розподілені за категоріями:

- *Розклади занять*: файли з інформацією про час, аудиторії, предмети та групи.
- *Лекції та навчальні матеріали*: документи, які містять лекційні записи, презентації, методичні посібники.
- *Контактна інформація*: список викладачів, адміністративних працівників та їхніх контактів.

2. *Формати файлів.*

Чат-бот повинен працювати з файлами різних форматів:

- *Текстові файли*: doc, docx, pdf.
- *Електронні таблиці*: xlsx для зберігання розкладу чи списків студентів.

3. *Логіка доступу до даних.*

Реалізується механізм індексації даних за допомогою інструменту *Llama Index*. Цей процес включає:

- Попередню обробку тексту (вилучення даних із документів).
- Побудову індексів для швидкого пошуку.

4. *Синхронізація даних.*

Чат-бот синхронізує дані з GitHub-репозиторієм, що дозволяє адміністраторам університету легко оновлювати інформацію. Кожен файл у репозиторії має зрозумілу структуру, яка полегшує індексацію.

3.2 Розробка алгоритму

3.2.1 Архітектура алгоритму: основні модулі та етапи.

Алгоритм роботи чат-бота побудований на модульній архітектурі, яка забезпечує його гнучкість, масштабованість і зручність інтеграції з корпоративними системами. Основні етапи алгоритму включають:

1. *Завантаження та оновлення даних:*

Як показано на рисунку 3.1 чат-бот регулярно синхронізує інформацію з GitHub [36]-репозиторію, включаючи файли розкладів, лекцій і списків студентів.

```python
def download_files_from_github(repo_url, local_path):
    response = requests.get(repo_url)
    files = response.json()
    for file in files:
        file_url = file['download_url']
        local_file_path = os.path.join(local_path, file['name'])
        with open(local_file_path, 'wb') as f:
            f.write(requests.get(file_url).content)
```

Рис. 3.1. Синхронізація з GitHub-репозиторієм

2. *Індексація даних:*

Для забезпечення швидкого доступу до інформації дані індексуються за допомогою *Llama Index [37]*, що дозволяє обробляти запити в реальному часі.

3. *Обробка запитів користувачів*:

Кожен запит аналізується за допомогою моделей обробки природної мови (NLP), щоб визначити намір користувача та ключові слова.

4. *Пошук релевантної інформації:*

На основі результатів індексації бот знаходить відповідну інформацію у проіндексованих файлах і готує відповідь.

5. *Генерація та надання відповіді:*

Відповідь на запит формується у вигляді тексту або посилання на файл, який містить потрібну інформацію.

3.2.2 Опис процесу завантаження даних із GitHub-репозиторію.

Завантаження даних із GitHub є критично важливим для роботи чат-бота, адже саме звідти надходить інформація, яка стає доступною для користувачів.

- *Підключення до репозиторію*

Використовується API GitHub для доступу до файлів у визначених директоріях, наприклад, schedule/ для розкладів і lectures/ для навчальних матеріалів.

- *Завантаження нових файлів*

Алгоритм перевіряє стан репозиторію на наявність нових або оновлених файлів. Якщо такі виявляються, вони завантажуються у локальну базу даних.

- *Зберігання даних*

Усі завантажені файли зберігаються в локальних папках, які відповідають категоріям інформації. Це дозволяє полегшити подальшу індексацію.

3.2.3 Побудова індексації для швидкого пошуку інформації.

Після завантаження даних із GitHub вони проходять процес індексації. Індексація забезпечує швидкий доступ до потрібної інформації за ключовими словами або фразами.

Етапи індексації:

1. *Обробка тексту*

Для кожного файлу текст вилучається, очищується від зайвих символів і приводиться до єдиного формату.

2. *Створення індексу*

Використовується *Llama Index,* що дозволяє зберігати витягнуту інформацію у вигляді структурованих даних.

3. *Оновлення індексу*

Кожного разу, коли в репозиторії додаються нові файли, індекс оновлюється, щоб включати ці зміни.

3.2.4 Обробка текстових запитів користувача.

Для аналізу текстових запитів бот використовує технології обробки природної мови (NLP). Моделі NLP дозволяють зрозуміти намір користувача (intent) та витягнути з тексту сутності (entity), які допомагають у формуванні відповіді.

Етапи обробки запиту:

1. *Попередня обробка*

Текст запиту очищується від зайвих символів і розбивається на окремі слова або фрази.

2. *Аналіз намірів*

Використовуючи NLP-моделі, бот визначає тип запиту, наприклад:

- Пошук розкладу.
- Пошук лекцій.
- Пошук контактної інформації.

3. *Формування запиту до індексу*

На основі аналізу запиту формується запит до індексу. Наприклад, запит "Знайди лекцію з програмування" спрямовується до індексу lectures.

3.2.5 Генерація відповідей і надання посилань на файли.

Після виконання пошуку бот формує відповідь, яка надсилається користувачеві. Відповідь може містити:

1. Текстову інформацію (наприклад, список занять для конкретного дня).
2. Посилання на файл, що відповідає запиту.
3. Графічну або візуальну інформацію (наприклад, діаграми, зображення або карти), що допомагає краще зрозуміти запитувану тему.
4. Інтерактивні елементи (наприклад, кнопки або форми для подальшої взаємодії), які дозволяють користувачу продовжити діалог або виконати додаткові дії.

3.3 Програмна інтеграція

Важливим кроком у розгортанні чат-бота є інтеграція програмного забезпечення, яка гарантує зв'язок між ботом і внутрішніми системами університету, такими як база даних, API зовнішніх платформ і системи обробки запитів. Інструменти, процедури та методи інтеграції, які гарантують надійність та корисність розробленого рішення, розглядаються в цьому розділі.

3.3.1 Вибір інструментів і технологій (LangChain, Llama Index, GitHub API).

Для реалізації програмної інтеграції були обрані такі інструменти:

1. *LangChain [38]* — бібліотека для управління діалогами та створення послідовностей запитів до моделей NLP.
2. *Llama Index* — інструмент для побудови ефективного індексу файлів різних форматів, що дозволяє пришвидшити пошук інформації.
3. *GitHub API* — забезпечує зручну синхронізацію даних із репозиторієм.
4. *Python [39]* — мова програмування завдяки її гнучкості та широкій екосистемі бібліотек (Requests, PyPDF2, Pandas тощо).
5. *Google Colab [40]* — потужне хмарне середовище для роботи з Jupyter Notebook, яке дозволяє програмістам і дослідникам зручно проводити обчислення, аналіз даних, машинне навчання та інші завдання прямо в хмарі, не витрачаючи час на налаштування власної інфраструктури.
6. Інтеграція цих технологій дозволила створити зручний і масштабований чат-бот.

3.3.2 Інтеграція з NLP-моделями для розпізнавання намірів користувачів.

Для аналізу тексту та визначення намірів користувачів використовуються моделі обробки природної мови (NLP). Це дозволяє чат-боту "розуміти" запити користувачів і повертати релевантні відповіді.

Процес інтеграції:

1. Використання моделей, таких як OpenAI GPT або spaCy, для розпізнавання ключових слів і класифікації запитів.

2. Реалізація класифікатора намірів (Intent Classifier) для маршрутизації запитів до відповідних модулів.

3. Інтеграція NLP-бібліотек у структуру бота через LangChain, що забезпечує обробку багатоступеневих діалогів.

3.3.3 Підключення внутрішніх систем компанії через API.

Для забезпечення роботи чат-бота в корпоративному середовищі важливо інтегрувати його з іншими системами компанії, такими як LMS, CRM або бази даних розкладів.

Процес інтеграції:

1. *Підключення до API внутрішніх систем*

Наприклад, LMS може надати API для отримання розкладу, завдань або лекцій. Використання REST API дозволяє боту відправляти запити й отримувати структуровану інформацію.

2. *Захист доступу через автентифікацію*

Використовується OAuth2 або JWT для забезпечення безпечного доступу до внутрішніх систем.

3. *Взаємодія з базою даних*

Бот інтегрується з локальною або хмарною базою даних для отримання або оновлення інформації.

3.3.4 Забезпечення безпеки та конфіденційності даних.

Безпека даних є критичним аспектом, особливо якщо бот працює з персональною інформацією. Для цього впроваджено наступні заходи:

1. *Шифрування даних*

Усі файли, що завантажуються в репозиторій, і запити до API передаються через захищені протоколи (HTTPS).

2. *Автентифікація користувачів*

Кожен користувач проходить перевірку перед доступом до персоналізованих даних. Це може бути верифікація через одноразовий пароль (OTP) або інтеграція з LDAP.

3. *Обмеження доступу*

Для різних категорій користувачів (студенти, викладачі, адміністрація) впроваджено рольову модель доступу, яка обмежує можливості отримання певних даних.

3.4 Розробка програми

Для початку ми підключаємо базу даних (рис.3.2) яка міститиме внутрішні дані університету, такі як розклад, індивідуальний навчальний план, тощо. Також даний код буде перевіряти, чи є оновлення у файлах, та завантажує їх у локальну базу.

```
! git clone https://github.com/Lilovich/context_data.git

import requests
import os

def download_files_from_github(repo_url, local_path):
    response = requests.get(repo_url)
    files = response.json()
    for file in files:
        file_url = file['download_url']
        local_file_path = os.path.join(local_path, file['name'])
        with open(local_file_path, 'wb') as f:
            f.write(requests.get(file_url).content)
```

Рис. 3.2. Підключення до бази даних

Наступний етап це встановлення залежностей проєкту, який буде відповідати за інсталяцію всіх необхідних бібліотек (рис.3.3), які використовуються в проєкті для обробки тексту, роботи з NLP-моделями, індексації даних та інтеграції з API.

```
!pip install llama-index
!pip install openai
!pip install langchain langchain-core
!pip install langchain_community
!pip install langchain-openai
!pip install jedi
!pip install pycairo
!pip install python-docx
!pip install PyPDF2
!pip install docx2txt
!pip check
!pip install sentence-transformers
```

Рис. 3.3. Встановлення залежностей проєкту

Виконуємо імпорт ключових бібліотек та модулів, необхідних для роботи з даними, індексації, обробки документів і інтеграції з NLP-моделями, як показано на рис.3.4.

```
from llama_index.core import SimpleDirectoryReader, VectorStoreIndex
from sentence_transformers import SentenceTransformer
from llama_index.llms.openai import OpenAI
from PyPDF2 import PdfReader
from docx import Document
import docx2txt
from llama_index.core import Document
from llama_index.core import load_index_from_storage, StorageContext
from llama_index.core import Settings
from langchain_openai import ChatOpenAI
from IPython.display import display, Markdown
import os
```

Рис. 3.4. Імпорт ключових бібліотек та модулів

Для роботи з файлами різного типу, наприклад .txt, .docx чи .pdf, напишемо функцію load_documents (див. рис.3.5) яка допоможе нам з обробкою тексту з файлів.

```python
def load_documents(directory_path):
    """
    Завантажує документи з .txt, .docx та .pdf файлів у вказаній директорії та її підкаталогах.
    """
    documents = []

    # Рекурсивно проходимо по всіх файлах у директорії та підкаталогах
    for root, dirs, files in os.walk(directory_path):
        for file in files:
            file_path = os.path.join(root, file)
            # Обробка .txt файлів
            if file.endswith(".txt"):
                with open(file_path, "r", encoding="utf-8") as f:
                    text = f.read()
                documents.append(Document(text=text, metadata={"file_path": file_path}))

            # Обробка .docx файлів
            elif file.endswith(".docx"):
                text = docx2txt.process(file_path)
                documents.append(Document(text=text, metadata={"file_path": file_path}))

            # Обробка .pdf файлів
            elif file.endswith(".pdf"):
                pdf_reader = PdfReader(file_path)
                text = ""
                for page in pdf_reader.pages:
                    text += page.extract_text()
                documents.append(Document(text=text, metadata={"file_path": file_path}))

    return documents
```

Рис. 3.5. Приклад функції load_documents

Для створення індексу текстових документів із використанням локальної моделі ембедингів на основі SentenceTransformers напишемо функцію construct_index_local (див. рис. 3.6). Вона дозволить представити текстові дані у вигляді векторів, які потім можуть використовуватися для пошуку схожих документів, кластеризації чи інших операцій з обробки тексту. Функція буде приймати набір текстових документів як вхідні дані, після чого кожен документ буде перетворений у вектор за допомогою моделі SentenceTransformers. Ці вектори потім зберігаються в індексі, що дозволяє ефективно виконувати пошук схожих документів за допомогою метрик відстані, таких як косинусна подібність. Крім того, індекс можна оновлювати в процесі роботи, додаючи нові документи та зберігаючи їх векторні представлення. Така система забезпечує високошвидкісний доступ до схожих документів, що особливо важливо для великих баз даних.

```python
def construct_index_local(directory_path):
    """
    Створює індекс за допомогою локального ембединг-модуля SentenceTransformers.
    """
    # Завантажуємо документи з усіх підкаталогів
    documents = load_documents(directory_path)

    if not documents:
        print("Не знайдено файлів для обробки!")
        return

    print(f"Знайдено {len(documents)} документів для індексації.")

    # Використання локальної моделі для ембедингів
    model = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = [model.encode(doc.text) for doc in documents]

    # Створення індексу з локальними ембедингами
    index = VectorStoreIndex.from_documents(documents, embeddings=embeddings)

    # Зберігаємо індекс
    storage_context = index.storage_context
    storage_context.persist("index_storage")
    print("Індекс успішно створено та збережено у 'index_storage'.")

    return index
```

Рис. 3.6. Приклад функції construct_index_local

Функція ask_ai (див. рис. 3.7) реалізує інструмент для інтерактивної взаємодії з індексом, створеним на основі текстових документів. Її основна мета полягає у забезпеченні можливості виконання запитів користувача до проіндексованих даних, що дозволяє отримувати відповіді на основі змісту документів.

У процесі виконання функції:

1. Завантажується раніше створений індекс за допомогою виклику StorageContext та функції load_index_from_storage. Якщо виникає помилка, користувач отримує відповідне повідомлення із деталями проблеми.

2. Після успішного завантаження індексу налаштовується QueryEngine, який відповідає за обробку запитів до даних.

3. Користувач вводить запит у консольному режимі. Якщо команда відповідає слову "exit", програма завершує свою роботу, попередньо повідомивши користувача.

4. Введений запит обробляється за допомогою query_engine.query(), після чого виводиться отримана відповідь. У разі виникнення помилок під час виконання запиту, вони фіксуються, і користувач отримує повідомлення із деталями.

```python
def ask_ai():
    """
    Завантажує індекс і дозволяє користувачу ставити запитання.
    """
    # Завантажуємо збережений індекс
    try:
        storage_context = StorageContext.from_defaults(persist_dir="index_storage")
        index = load_index_from_storage(storage_context)
    except Exception as e:
        print("Помилка завантаження індексу! Перевірте, чи був створений індекс.")
        print(f"Деталі помилки: {e}")
        return

    print("Індекс успішно завантажено. Ви можете ставити запитання!")

    # Налаштовуємо QueryEngine
    query_engine = index.as_query_engine()

    while True:
        # Введення запиту користувача
        query = input("What do you want to ask? (або введіть 'exit' для виходу): ")
        if query.lower() == "exit":
            print("Дякую за використання!")
            break

        # Виконання запиту
        try:
            response = query_engine.query(query)
            print(f"**Відповідь:** {response}")
        except Exception as e:
            print(f"Сталася помилка під час виконання запиту: {e}")
```

Рис. 3.7. Приклад функції ask_ai

Як зображено на рисунку 3.8 спочатку я налаштовую середовище, встановлюючи API-ключ для доступу до сервісів OpenAI. Це необхідно для автентифікації та використання можливостей NLP-моделей, якщо вони знадобляться в подальшій роботі. Далі я викликаю функцію construct_index_local із шляхом до каталогу, де зберігаються текстові дані. У цьому випадку я використовую каталог context_data/dataBase, оскільки там знаходяться документи, які необхідно проіндексувати. Ця функція завантажує всі документи з вказаної директорії, перетворює їх у векторні представлення (ембединги) за допомогою локальної моделі

SentenceTransformers, а потім створює індекс. Індекс зберігається у локальному сховищі для подальшого використання.

```
os.environ["OPENAI_API_KEY"] = "sk-proj-87rQJ0uMyK-Z_t1JbXtPNLEOyoEgAFbx9o-_3dm

construct_index_local("context_data/dataBase")
```

Рис. 3.8. Встановлення API-ключа

Користувач може вводити запитання у текстовій формі, наприклад, запитувати розклад студентів (рис.3.9) на конкретний день чи інші деталі. Система обробляє запит, використовуючи створений індекс, і шукає відповідну інформацію серед збережених даних. Завдяки семантичному пошуку бот не лише знаходить точні збіги, а й аналізує контекст, забезпечуючи максимально релевантну відповідь.

```
ask_ai()

Індекс успішно завантажено. Ви можете ставити запитання!
What do you want to ask? (або введіть 'exit' для виходу): Виведи розклад ІПм-23-1 на 25.09
**Відповідь:** The schedule for group ІПм-23-1 on 25.09 is as follows:

- 08:00 to 10:50 -
- 11:00 to 12:20: *(в) Управління стартапами (Пр) Збірна група 14М21-3.3 (АКПм-23-2, ЕТМм-23-1, ІПм-23-1, ІПм-23-2, ІПм-23-3, ІСКм-23-1) доцент Станьковська Ірина Мирославівна
- 12:50 to 14:10: *(в) Тренінг-курс `Стартап і фандрейзинг в інноваційному бізнесі` (Пр) Збірна група 15М21-2.2 (ЕТСм-23-2, НІМм-23-1, ПТм-23-1, ІПм-23-1, ІПм-23-2, ІСКм-23-1) д
- 14:20 to 15:40: *(в) Логістика на автомобільному транспорті (Пр) Збірна група 12М21-1.1 (АКПм-23-1, АКПм-23-2, ПМЗм-23-1, ІПм-23-1, ІПм-23-2, ІПм-23-3) доцент Гнип Марія Михай
- 15:50 to 17:10: *(в) Архітектура обчислювальних систем (Лаб) Збірна група 19М21-1.1 (КІм-23-2, ІПм-23-1, ІПм-23-2) асистент Лазорів* Алла Миколаївна
- 17:20 to 18:40: -
```

Рис. 3.9. Запит на розклад

Окрім розкладу, можна зручно та швидко дізнатись які дисципліни ти вивчаєш, як продемонстровано на рисунку 3.10.

```
ask_ai()

Індекс успішно завантажено. Ви можете ставити запитання!
What do you want to ask? (або введіть 'exit' для виходу): Виведи індивідуальний план навчання студента Глушко Михайло Сергійович
**Відповідь:** Індивідуальний навчальний план студента Глушко Михайло Сергійович:
-Виконання та захист магістерської роботи
-Науково-дослідна практика з аналітики програмного забезпечення
-Науково-дослідний практикум з розробки та прийняття рішень
-Теоретичні та прикладні аспекти створення та функціонування мов програмування
*Вибірковий компонент
-*(в) Основи штучного інтелекту
-*(в) Інженерія сервісів і систем зберігання та обробки даних
What do you want to ask? (або введіть 'exit' для виходу):
```

Рис. 3.10. Запит на індивідуальний план навчання

3.5 Висновки до розділу

У третьому розділі ми створили та впровадили алгоритм чат-бота для автоматизації внутрішніх організаційних процедур. Основною метою було створення технологій, які дозволять швидко та ефективно обробляти запити користувачів - студентів, викладачів чи співробітників компанії. Для досягнення цієї мети важливо було розробити систему, яка здатна забезпечити інтуїтивно зрозумілий інтерфейс та швидку обробку запитів.

Частиною процесу розробки було виявлення важливих функцій чат-бота, які задовольняють очікування користувачів, таких як пошук розкладу, отримання навчальних матеріалів, пошук контактної інформації та персоналізованих даних. Для цього було використано модель локального вбудовування для побудови системи індексації текстових документів, яка дозволяє представляти текстові дані у вигляді векторів для швидкого пошуку та аналізу. Це значно підвищує точність і швидкість пошукових запитів, дозволяючи чат-боту ефективно знаходити потрібну інформацію навіть серед великого обсягу даних.

Крім того, було створено інтерактивну систему запитів, яка використовує семантичний пошук для полегшення залучення користувачів. Така система дозволяє чат-боту не тільки точно відповідати на запити, але й адаптуватися до різних формулювань запитів, що робить його більш зручним для користувачів з різним рівнем технічної підготовки.

У роботі також було враховано зв'язок програмного забезпечення з внутрішніми системами, включаючи бази даних, репозиторії GitHub та інші API, які дозволяють синхронізувати та оновлювати дані в реальному часі. Це дозволяє чат-боту отримувати актуальну інформацію з різних джерел, що покращує якість наданих відповідей і забезпечує підтримку користувачів на всіх етапах взаємодії.

Особливу увагу було приділено безпеці даних, впроваджено заходи з контролю доступу, аутентифікації користувачів та шифрування, що гарантує збереження конфіденційності та захищеність інформації.

ВИСНОВКИ

У цій магістерській роботі розглядаються теорії, методи та алгоритми інтеграції чат-ботів для автоматизації внутрішніх процесів компанії - важливого етапу цифрової трансформації бізнесу. У цій роботі проаналізовано сучасні технології розпізнавання природної мови, алгоритми діалогової взаємодії та стратегії інтеграції чат-ботів з бізнес-інформаційними системами. Особливу увагу приділено створенню архітектури, яка гарантує безперебійну взаємодію з такими програмами, як CRM, ERP та HRM, а також алгоритмам обробки запитів, маршрутизації та управління діалогами.

В результаті дослідження було запропоновано новий дизайн чат-бота, який може значно підвищити ефективність взаємодії користувачів з корпоративними системами. За допомогою запропонованих алгоритмів і моделей було створено прототип програмного забезпечення, який автоматизує повторювану роботу, мінімізує людську взаємодію та покращує комунікацію між співробітниками. Система використовує сучасні криптографічні протоколи та алгоритми, включаючи TLS, RSA та AES, що гарантує безпеку та надійність даних.

Експерименти показали, що розроблена система є високоефективною для автоматизації робочих місць і відповідає встановленим специфікаціям. Потенціал використання запропонованих методів у різних сферах, включаючи планування ресурсів, технічну допомогу, управління персоналом та документообіг, надає створеному рішенню практичної цінності.

Таким чином, результати дослідження підтверджують, що інтеграція чат-ботів - це ефективний спосіб оптимізувати роботу, скоротити витрати та підвищити загальну корпоративну ефективність. Висновки дослідження можуть бути використані як основа для подальшого розвитку технологій автоматизації та використання в корпоративному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bourbakis N. G. Artificial intelligence and automation / Nikolas G. Bourbakis. – [S. l.] : WORLD SCIENTIFIC, 1998.

2. Ethics M. ELIZA: the accidental chatbot that shaped AI history | hackernoon / Machine Ethics // HackerNoon - read, write and learn about any technology.

3. Contributors to Wikimedia projects. PARRY - wikipedia [Electronic resource] / Contributors to Wikimedia projects // Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/PARRY.

4. Dialogflow [Electronic resource] // Dialogflow. – Mode of access: https://dialogflow.cloud.google.com/.

5. Microsoft Bot Framework [Electronic resource] // Microsoft Bot Framework. – Mode of access: https://dev.botframework.com/. – Title from screen.

6. Amazon Lex [Electronic resource] // Amazon Lex. – Mode of access: https://aws.amazon.com/lex/.

7. *IBM Watson Assistant* [Electronic resource] // *IBM Watson Assistant*. – Mode of access: https://www.ibm.com/products/watsonx-assistant.

8. BotPenguin. Mitsuku Chatbot Tutorial: How to use Mitsuku Chatbot in 2023 [Electronic resource] / BotPenguin // Medium. – Mode of access: https://botpenguin.medium.com/mitsuku-chatbot-tutorial-how-to-use-mitsuku-chatbot-in-2023-fefeebb37418.

9. Telegram Bot API [Electronic resource] // Telegram APIs. – Mode of access: https://core.telegram.org/bots/api.

10. Messenger-Plattform - Dokumentation - Meta for Developers [Electronic resource] // Social Technologies | Meta for Developers. – Mode of access: https://developers.facebook.com/docs/messenger-platform/.

11. Що таке NLP (обробка природної мови)? [Електронний ресурс] // Unite.AI. – Режим доступу: https://www.unite.ai/uk/what-is-natural-language-processing/.

12. Natural language processing and information systems [Electronic resource] / ed. by R. Muñoz, A. Montoyo, E. Métais. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – Mode of access: https://doi.org/10.1007/978-3-642-22327-3 (date of access: 24.12.2024).

13. NLTK Tokenize: Токенізатор слів і речень із прикладом. Guru99. URL: https://www.guru99.com/uk/tokenize-words-sentences-nltk.html

14. Bondarenko O. Словник NLP. Medium. URL: https://medium.com/stinopys/словник-nlp-b0fab1027551#97c8

15. Stopwords. RANKS NL. URL: https://www.ranks.nl/stopwords

16. Що таке лематизація в тексті | Секрети від копірайтера | Fabrika Slov. Контент-студія «Fabrika Slov». URL: https://fabrika-slov.com/uk/chto-takoe-lemmatizacziya/

17. Building scalable database applications: Object-oriented design, architectures, and implementations. – Reading, Mass : Addison-Wesley, 1998. – 311 p.

18. Що таке стемінг у копірайтингу? | FabrikaSlov. Контент-студія «Fabrika Slov». URL: https://fabrika-slov.com/uk/chto-takoe-stemming/

19. Вбудовування Word і модель Word2Vec із прикладом. Guru99. URL: https://www.guru99.com/uk/word-embedding-word2vec.html

20. Алгоритм BERT: що вміє та не вміє робити. WEDEX. URL: https://wedex.com.ua/blog/algoritm-bert-shho-vmiye-ta-ne-vmiye-robiti/

21. Аналіз настроїв: визначення, типи, випадки використання, важливість. Shaip. URL: https://uk.shaip.com/blog/the-what-why-and-how-of-sentiment-analysis/

22. Андрій Денисенко. Вступ до REST API – RESTful вебсервіси. robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс. URL: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi

23. Аутентифікація та авторизація: Захист даних користувачів. IT Рейтинг України. URL: https://it-rating.ua/autentifikatsiya-ta-avtorizatsiya-zahist-danih-koristuvachiv

24. Oauth що це: визначення технології та основні принципи роботи. FoxmindEd. URL: https://foxminded.ua/oauth-shcho-tse/

25. Bishop B. Dancing with robots: the 29 strategies for success in the age of AI and automation / Bill Bishop. – [S. l.] : Dundurn Press, 2022. – 200 p.

26. Milanović D. M. How Does OAuth 2.0 Work. Medium. URL: https://medium.com/@techworldwithmilan/how-does-oauth-2-0-work-bea67a760aa5

27. Віддалене виконання коду в JsonWebToken - CoreWin. CoreWin. URL: https://corewin.ua/blog/jwt-analysis/

28. Chowdhury S. JSON Web Token (JWT) – The right way of implementing, with Node.js. Medium. URL: https://siddharthac6.medium.com/json-web-token-jwt-the-right-way-of-implementing-with-node-js-65b8915d550e

29. Головна - Репозитарій Вінницького Національного Технічного Університету.
URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/39823/12461.pdf?sequence=3

30. Система кешування в мережі: Вплив на продуктивність та доступність контенту. PROXY RATING. URL: https://proxys-rating.com/systema-keshuvannya-v-merezhi-vplyv-na-produktyvnist-ta-dostupnist-kontentu/

31. Робота з базами даних в Python: SQL та NoSQL рішення - Блог Mate academy. Блог Mate academy. URL: https://mate.academy/blog/python/sql-nosql-databases-python/

32. Rudresh Narwal. SQL vs NoSQL Databases. Medium. URL: https://medium.com/@rudresh.narwal/sql-vs-nosql-databases-1bd43d69adbc

33. Типи чат-ботів, як вони працюють, варіанти створення чат-бота. Платформа для навчання у месенджерах. URL: https://lessondelivery.org/chatbot/robota-chat-bota-sposoby-stvorennia-koryst.html

34. Redirecting. Home Page. URL: https://doi.org/10.1016/j.neucom.2019.08.007

35. Minton S. Journal of Artificial Intelligence Research (Journal of Artificial Intelligence) / Steven Minton. – [S. l.] : Morgan Kaufmann Pub, 1996. – 600 p.

36. GitHub · Build and ship software on a single, collaborative platform [Electronic resource] // GitHub. – Mode of access: https://github.com.

37. LlamaIndex - LlamaIndex [Electronic resource] // LlamaIndex - LlamaIndex. – Mode of access: https://docs.llamaindex.ai/en/stable/.

38. AI and machine learning. – [S. l.] : SAGE Publications India Pvt, Ltd., 2020. – 180 p.

39. LangChain [Electronic resource] // LangChain. – Mode of access: https://www.langchain.com. – Title from screen.

40. Welcome to python.org [Electronic resource] // Python.org. – Mode of access: https://www.python.org/doc/.

41. Як налаштувати середовище виконання в Google Colab: повний гід [Електронний ресурс] // FoxmindEd. – Режим доступу: https://foxminded.ua/google-colab/.

ДОДАТКИ

Додаток А

Фрагменти програмних лістингів

```
! git clone https://github.com/Lilovich/context_data.git
import requests
import os

def download_files_from_github(repo_url, local_path):
    response = requests.get(repo_url)
    files = response.json()
    for file in files:
        file_url = file['download_url']
        local_file_path = os.path.join(local_path, file['name'])
        with open(local_file_path, 'wb') as f:
            f.write(requests.get(file_url).content)

!pip install llama-index
!pip install openai
!pip install langchain langchain-core
!pip install langchain_community
!pip install langchain-openai
!pip install jedi
!pip install pycairo
!pip install python-docx
!pip install PyPDF2
!pip install docx2txt
!pip check
!pip install sentence-transformers

from llama_index.core import SimpleDirectoryReader,
VectorStoreIndex
from sentence_transformers import SentenceTransformer
from llama_index.llms.openai import OpenAI
from PyPDF2 import PdfReader
```

```
from docx import Document
import docx2txt
from llama_index.core import Document
from llama_index.core import load_index_from_storage,
StorageContext
from llama_index.core import Settings
from langchain_openai import ChatOpenAI
from IPython.display import display, Markdown
import os

def load_documents(directory_path):
    """
    Завантажує документи з .txt, .docx та .pdf файлів у вказаній
директорії та її підкаталогах.
    """
    documents = []

      # Рекурсивно проходимо по всіх файлах у директорії та
підкаталогах
    for root, dirs, files in os.walk(directory_path):
        for file in files:
            file_path = os.path.join(root, file)
            # Обробка .txt файлів
            if file.endswith(".txt"):
                with open(file_path, "r", encoding="utf-8") as f:
                    text = f.read()
                                documents.append(Document(text=text,
metadata={"file_path": file_path}))

            # Обробка .docx файлів
            elif file.endswith(".docx"):
                text = docx2txt.process(file_path)
                                documents.append(Document(text=text,
metadata={"file_path": file_path}))
```

```
                    # Обробка .pdf файлів
                    elif file.endswith(".pdf"):
                        pdf_reader = PdfReader(file_path)
                        text = ""
                        for page in pdf_reader.pages:
                            text += page.extract_text()
                                    documents.append(Document(text=text,
metadata={"file_path": file_path}))

            return documents

        def construct_index_local(directory_path):
            """
              Створює індекс за допомогою локального ембединг-модуля
SentenceTransformers.
            """
            # Завантажуємо документи з усіх підкаталогів
            documents = load_documents(directory_path)

            if not documents:
                print("Не знайдено файлів для обробки!")
                return

            print(f"Знайдено {len(documents)} документів для індексації.")

            # Використання локальної моделі для ембедингів
            model = SentenceTransformer("all-MiniLM-L6-v2")
            embeddings = [model.encode(doc.text) for doc in documents]

            # Створення індексу з локальними ембедингами
                 index    =    VectorStoreIndex.from_documents(documents,
embeddings=embeddings)

            # Зберігаємо індекс
            storage_context = index.storage_context
```

```
        storage_context.persist("index_storage")
            print("Індекс успішно створено та збережено у 'index_storage'.")

        return index

    def ask_ai():
        """
        Завантажує індекс і дозволяє користувачу ставити запитання.
        """
        # Завантажуємо збережений індекс
        try:
                                            storage_context = StorageContext.from_defaults(persist_dir="index_storage")
            index = load_index_from_storage(storage_context)
        except Exception as e:
            print("Помилка завантаження індексу! Перевірте, чи був створений індекс.")
            print(f"Деталі помилки: {e}")
            return

        print("Індекс успішно завантажено. Ви можете ставити запитання!")

        # Налаштовуємо QueryEngine
        query_engine = index.as_query_engine()

        while True:
            # Введення запиту користувача
            query = input("What do you want to ask? (або введіть 'exit' для виходу): ")
            if query.lower() == "exit":
                print("Дякую за використання!")
                break
```

```
            # Виконання запиту
            try:
                response = query_engine.query(query)
                print(f"**Відповідь:** {response}")
            except Exception as e:
               print(f"Сталася помилка під час виконання запиту: {e}")

    os.environ["OPENAI_API_KEY"] = "sk-proj-87rQJ0uMyK-
Z_t1JbXtPNLEOyoEgAFbx9o-_3dmtgLmpMApbqH2p2sd9a2dO4BFR2Bj24iTQ-
UT3BlbkFJnhJo7JQ09bGgm3-
fPWiZZPd9E2JNCR7_S9gdxwJuvO0QMvA_5y1YPRlLUZjCOmMECwTZYwTM8A"

    construct_index_local("context_data/dataBase")

    ask_ai()
```