

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Розробка сайту для лікарні з чат-ботом (з елементами ШІ)»

Виконала студентка групи ПЗм-22-1
СКРІПЧЕНКО Дар'я Геннадіївна

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент
МОРОЗОВ Андрій Васильович

Житомир – 2023

РЕФЕРАТ

Індивідуальне завдання кваліфікаційної роботи полягає в дослідженні та обґрунтуванні вибору моделей, методів та технологій, які будуть використанні для подальшої розробки чат-боту та інформаційного веб-ресурсу для лікарні.

Кваліфікаційна магістерська робота на тему «Розробка сайту для лікарні з чат-ботом (з елементами ШІ)» складається з теоретичного та практичного комплексу системи та пояснювальної записки. Пояснювальна записка до кваліфікаційної магістерської роботи складається зі вступу, трьох розділів, списку літератури з 41 найменування, 30 рисунків та 3 додатків. Повний обсяг роботи становить 67 сторінок.

КЛЮЧОВІ СЛОВА: ШІ, ЧАТ-БОТ, КЛІНІКА, ПОЛІКЛІНІКА, ГЛИБОКЕ НАВЧАННЯ, РОЗУМІННЯ ПРИРОДНОЇ МОВИ, NTLK, DNN, NLU, PYTHON, TENSORFLOW, TFLEARN, CHATTERBOT.

ABSTRACT

The individual task of the qualification work is to research and justify the choice of models, methods and technologies that will be used for the further development of a chatbot and an informational web resource for the hospital.

The qualifying master's thesis on the topic "Development of a website for a hospital with a chatbot (with AI elements)" consists of a theoretical and practical complex of the system and an explanatory note. The explanatory note to the qualifying master's thesis consists of an introduction, three chapters, a bibliography of 41 names, 30 figures and 3 appendices. The full volume of the work is 67 pages.

KEYWORDS: AI, CHATBOT, CLINIC, POLYCLINIC, DEEP LEARNING, NATURAL LANGUAGE UNDERSTANDING, NTLK, DNN, NLU, PYTHON, TENSORFLOW, TFLEARN, CHATTERBOT.

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ	
ВИРІШЕННЯ ЗАДАЧ.....	8
1.1 Аналіз задачі, засобів та методів її вирішення	8
1.1.1 Концепції, методи та алгоритми розробки сайтів.....	8
1.1.2 Концепції, методи та алгоритми розробки чат-ботів.....	10
1.1.3 Основи концепції, поняття та алгоритми роботи DNN	15
1.2 Аналіз наявних веб-ресурсів лікарень та поліклінік.....	18
1.3 Обґрунтування обраних технологій і інструментів	21
Висновки до розділу 1	29
РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТА КОНЦЕПЦІЙ	
РОЗРОБКИ ЧАТ-БОТУ	30
2.1 Розробка чат-боту без використання фрейму ChatterBot.....	30
2.1.1 Розробка чат-боту на основі DNN	30
2.1.2 Аналіз роботи чат-боти та його результати в діаграмах і графіках.....	38
2.2 Розробка чат-боту за допомогою фрейму ChatterBot.....	41
Висновки до розділу 2	44
РОЗДІЛ 3. РОЗГОРТАННЯ ВЕБ-РЕСУРСУ ТА АНАЛІЗ НЕОБХІДНОСТІ	
ВИКОРИСТАННЯ ЧАТ-БОТІВ.....	45
3.1 Розгортання веб-ресурсу.....	45
3.2 Аналіз необхідності використання чат-ботів	48
Висновки до третього розділу	53
ВИСНОВКИ	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТКИ	61

ВСТУП

Актуальність теми. Хоча медицина тривалий час розглядалася як професія, що базується виключно на живому спілкуванні, останнім часом стає все більш очевидним, що використання ІТ загалом, та зокрема електронних комунікацій, забезпечує не тільки економічну ефективність діяльності в медичній сфері, але й також спосіб забезпечення дистанційного зв'язку між медичним персоналом і пацієнтами або особами, які за ними доглядають.

Використання Інтернету в лікувальній діяльності стало реальністю останніх років, а телемедицина або віртуальна лікарня перейшли етап наукової фантастики в реальність [20]. Розвиток інформаційного суспільства, збільшення кількості людей, які мають доступ до комп'ютера, навіть людей похилого віку (включаючи розробку для них вступних курсів з комп'ютера), диверсифікація інформації та необхідність якнайшвидшої взаємодії зумовили адаптацію медичних підрозділів до засоби передачі інформації, щоб зробити їх більш привабливими, простішими в експлуатації та використанні. Розробка систем із гарантованою якістю інформації має бути постійною турботою спеціалістів у сфері громадського здоров'я, які мають працювати разом із ІТ-фахівцями та створювати власні веб-сторінки.

Багато інтернет-сайтів намагаються зарекомендувати себе як джерела медичної інформації та засоби для зв'язку пацієнтів з лікарями [20]. Кількість відвідувачів цих сайтів, і величезна кількість сайтів свідчить про громадськість інтерес до цих послуг. Тому їм потрібно використовувати наявні технології для ефективного спілкування з ними пацієнтів, а також продовжувати покращувати їх і створювати нові.

Розробка та вдосконалення веб-сайтів клінік і лікарень має бути в центрі уваги фахівців у сфері охорони здоров'я та інформатики. Веб-сайти клінік та лікарень повинні не тільки забезпечувати дотримання етичних правил, бути функціональними та професійними з точки зору інформатики, але й надавати адекватну інформацію потенційним клієнтам медичних підрозділів. Веб-сайти

– це не лише засіб спілкування, але й зручний ІТ-інструмент для кращого управління цими структурами.

Мета і задачі дослідження. Метою дослідження є аналіз методів та засобів для розробки якісного та зручного для користувача інформаційного ресурсу для клініки/лікарні шляхом розробки чат-боту для допомоги з навігацією.

Для досягнення даної мети необхідно вирішити наступні задачі:

- 1) проаналізувати наявні веб-ресурси клінік та лікарень, обґрунтувати вибір напрямку та інструментів розробки;
- 2) проаналізувати наявні інструменти для розробки чат-ботів та обґрунтувати вибір інструментів;
- 3) визначити архітектуру та узагальнену структуру веб-сайту, спроектувати та реалізувати структурні складові та алгоритми роботи веб-сайту;
- 4) визначити архітектуру та узагальнену структуру чат-боту, спроектувати та реалізувати структурні складові та алгоритми роботи чат-боту;
- 5) реалізувати інструмент для розробки чат-боту та провести його тренування;
- 6) провести аналіз необхідності використання чат-ботів при розробці веб-ресурсів.

Об’єкт дослідження – процес розробки зручного для використання веб-ресурсу, зокрема розробки чат-боту для допомоги з навігацією розробленим веб-сайтом.

Предмет дослідження – методи і засоби розробки зручного для використання веб-ресурсу, зокрема розробки чат-боту для допомоги з навігацією розробленим веб-сайтом.

Практичне значення роботи полягає у наступному:

- розроблено веб-ресурс для поліклініки, який може бути оновлений та застосований для будь-якої лікарні / поліклініки;

- розроблено чат-бот, що допомагає з навігацією розробленим веб-ресурсом, та може бути оновленим та застосованим для інших подібних веб-ресурсів.

Матеріали дослідження за тематикою кваліфікаційної роботи доповідалися на V Всеукраїнській науково-технічній конференції "Комп'ютерні технології: інновації, проблеми, рішення" 01-02 грудня 2022 р. [1], III Всеукраїнській науково-практичній Інтернет-конференції «Сучасні комп'ютерні та інформаційні системи і технології» 12-19 грудня 2022 р. [2] та XIII Міжнародній науково-технічній конференції 30-31 березня 2023 р. [3]

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧ

1.1. Концепції, методи та алгоритми розробки веб-ресурсу з чат-ботом

1.1. 1. Концепції, методи та алгоритми розробки сайтів

Ефективний дизайн веб-сайту повинен виконувати свою призначену функцію, виражаючи своє конкретне повідомлення та одночасно залучаючи відвідувача. Кілька факторів, як, наприклад, послідовність, кольори, шрифти, зображення, простота та функціональність, сприяють гарному дизайну веб-сайту [27].

Під час розробки веб-сайту є багато ключових факторів, які впливатимуть на те, як він сприймається. Добре розроблений веб-сайт може допомогти викликати довіру у користувачів та скерувати їх до дій. Створення чудової взаємодії з користувачем передбачає переконання, що дизайн веб-сайту оптимізовано для зручності використання (форма та естетика) і наскільки простий у використанні (функціональність).

Веб-сайт повинен відповідати потребам користувача. Наявність простого чіткого наміру на всіх сторінках допоможе користувачеві взаємодіяти з тим, що пропонує веб-сайт.

Простота – це найкращий шлях, якщо враховувати взаємодію з користувачем і зручність веб-сайту [27].

- Кольорова гама

Колір має силу передавати повідомлення та викликати емоційні реакції. Ось чому важливо знайти колірну палітру, яка б відповідала темі веб-сайту. Краще обмежити вибір кольорів до п'яти кольорів. Приємні поєднання кольорів сприяють залученню клієнтів і створюють відчуття комфорту.

- Шрифти

Шрифт відіграє важливу роль на веб-сайті. Він привертає увагу та працює як візуальна інтерпретація теми веб-сайту. Шрифти мають бути

розбірливими, а також краще використовувати не більше трьох різних шрифтів на веб-сайті.

- **Зображення**

Зображення – це кожен візуальний аспект, який використовується в комунікації. Це включає фотографії, ілюстрації, відео та всі форми графіки. Усі образи мають бути виразними та передавати дух компанії та бути втіленням індивідуальності бренду. Більшість початкової інформації, яку ми отримуємо на веб-сайтах, є візуальною, і для першого враження важливо використовувати високоякісні зображення, щоб сформувати у відвідувачів враження про професіоналізм і довіру.

Шаблон на основі F – це найпоширеніший спосіб, у який відвідувачі сканують текст на веб-сайті. Дослідження відстеження очей показали, що більшість того, що люди бачать, знаходиться у верхній і лівій частинах екрана. Розклад у формі букви F імітує поширений спосіб читання зліва направо та зверху вниз. Даний шаблон сканування сторінки допоможе користувачам легше орієнтуватися на веб-сторінці.

Візуальна ієрархія — це розташування елементів у порядку важливості. Це робиться за розміром, кольором, зображеннями, контрастністю, шрифтами, пробілами, текстурою та стилем. Однією з найважливіших функцій візуальної ієрархії є встановлення фокусу; це показує відвідувачам, де знаходиться найважливіша інформація.

Сітки допомагають структурувати дизайн і організовувати вміст. Сітка допомагає вирівнювати елементи на сторінці та підтримувати її чистотою. Розмітка на основі сітки впорядковує вміст у чітку жорстку сіткову структуру зі стовпцями, розділами, які вирівнюються та відчують збалансованість, надають порядок і створюють естетично привабливий веб-сайт.

Більше людей використовують свої телефони чи інші пристрої для перегляду веб-сторінок. Важливо подумати про створення веб-сайту з адаптивним макетом, який може адаптуватися до різних екранів [27].

Навігація – це система орієнтування, яка використовується на веб-сайтах, де відвідувачі взаємодіють і знаходять те, що шукають. Навігація веб-сайтом є ключем до утримання відвідувачів [27]. Якщо навігація веб-сайту заплутана, відвідувачі можуть здатися та віднайти те, що їм потрібно, в іншому місці. Важливо, щоб навігація була простою, інтуїтивно зрозумілою та послідовною на кожній сторінці. Впровадження чат-бота може допомогти з навігацією веб-сайта, щоб відвідувачі не тільки легше дізнавалися необхідну їм інформацію, але й завжди отримували відповіді на свої запитання.

1. 1. 2. Концепції, методи та алгоритми розробки чат-ботів

Чат-бот – комп'ютерна програма, розроблена на основі нейромереж та технологій машинного навчання, за допомогою якої можливо здійснювати комунікацію в аудіо- або текстовому форматі. Чат-бот використовують для виконання конкретних завдань (наприклад, отримання довідкової інформації, виконання розрахунків) або задля розваги [16].

У 1950 році була опублікована відома стаття Алана Тюрінга «Обчислювальна техніка та інтелект», в якій запропоновано те, що зараз називається тестом Тюрінга, як критерій інтелекту. Цей критерій залежить від здатності комп'ютерної програми видавати себе за людину під час письмової розмови в режимі реального часу з людиною-суддею настільки, наскільки суддя не в змозі надійно відрізнити – лише на основі вмісту розмови – між програмою та справжньою людиною. Погана популярність запропонованого Тюрінгом тесту викликала великий інтерес до програми Джозефа Вайзенбаума ELIZA, опублікованої в 1966 році, яка, здавалося, могла обдурити користувачів, щоб вони повірили, що вони розмовляють зі справжньою людиною [5].

Основний метод роботи ELIZA (з тих пір скопійований розробниками чат-ботів) передбачає розпізнавання ключових слів або фраз у вхідних даних і виведення відповідних заздалегідь підготовлених або запрограмованих відповідей, які можуть рухати розмову вперед у очевидно значимий спосіб. (наприклад, відповідаючи на будь-який вхід, який містить слово «МАМА»,

«РОЗКАЖИ МЕНІ БІЛЬШЕ ПРО СВОЮ СІМ'Ю»). Таким чином створюється ілюзія розуміння, навіть якщо обробка була просто поверхневою. ELIZA показала, що таку ілюзію напрочуд легко створити, оскільки людські судді настільки готові віддавати перевагу сумнівам, коли розмовні відповіді можна витлумачити як «розумні».

Розробники інтерфейсів зрозуміли, що готовність людей сприймати комп'ютерні дані як справжню розмову – навіть якщо вони насправді базуються на досить простому зіставленні шаблонів – можна використати для корисних цілей [5]. Більшість людей вважають за краще працювати з програмами, схожими на людину, і це надає методам у стилі чат-ботів потенційно корисну роль в інтерактивних системах, яким потрібно отримувати інформацію від користувачів, якщо ця інформація є відносно простою та потрапляє в передбачувані категорії. Таким чином, наприклад, системи онлайн-довідки можуть з користю використовувати методи чат-ботів для визначення області допомоги, яка потрібна користувачам, потенційно забезпечуючи «дружніший» інтерфейс, ніж більш формальний пошук або система меню. Таке використання дає перспективу переміщення технології чат-ботів з «полиці... зарезервованої для курйозів» Вайзенбаума на позначену «справді корисні обчислювальні методи».

Останні відомі програми включають A.L.I.C.E., Jabberwacky і D.U.D.E (Agence Nationale de la Recherche і CNRS 2006). У той час як ELIZA та PARRY використовувалися виключно для симуляції текстової розмови, багато чат-ботів тепер включають інші функціональні функції, такі як ігри та можливості пошуку в Інтернеті. У 1984 році була опублікована книга під назвою *The Policeman's Beard is Half Constructed*, нібито написана чат-ботом Racter (хоча випущена програма не була б здатна це зробити).

З 1978 року до деякого часу після 1983 року проект CYRUS під керівництвом Джанет Колоднер створив чат-бота, що імітує Сайруса Венса (57-го держсекретаря США). Він використовував міркування на основі випадків і щодня оновлював свою базу даних, аналізуючи електронні новини

від United Press International. Програма не змогла обробити новини після раптової відставки Сайруса Венса в квітні 1980 року, і команда створила іншого чат-бота, який імітував його наступника, Едмунда Маскі.

Зазвичай, чат-бот пропонує користувачу обрати варіант серед запропонованих. Тобто отримання інформації або виконання інших задач відбувається шляхом вибору запропонованих відповідей, або запропонованих категорій, і користувач не вводить текст до вікна чату. Більш складні боти надають можливість вводу тексту і отримування інформації відповідно до запиту користувача [32].

Віртуальні чат-боти дозволяють робити замовлення або спілкуватися в режимі онлайн.

Чат-боти також з'являються в галузі охорони здоров'я. Дослідження показало, що лікарі в Сполучених Штатах вважають, що чат-боти будуть найбільш корисними для запису на прийом до лікаря, визначення місцезнаходження клінік або надання інформації про ліки.

Whatsapp об'єднався зі Всесвітньою організацією охорони здоров'я (ВООЗ), щоб створити сервіс чат-бота, який відповідає на запитання користувачів щодо COVID-19.

У 2020 році уряд Індії запустив чат-бот під назвою MyGov Corona Helpdesk, який працював через Whatsapp і допомагав людям отримувати інформацію про пандемію коронавірусу (COVID-19).

За сферою застосування чат-боти поділяють на:

- р2р — персональні комунікації (для особистого спілкування);
- b2c — споживчі (підтримка клієнтів компанії на корпоративному сайті та в мобільних додатках).

Чат-боти в основному використовують штучний інтелект для спілкування з користувачами, тому надають релевантний контент і актуальні пропозиції. Вони функціонують на основі набору інструкцій або використовують машинне навчання.

Певними основними концепціями, які пов'язані з технологією чат-ботів, є наступні [4]:

- Зіставлення зі зразком (Pattern matching), що базується на типових блоках стимул-реакція. Вводиться речення (стимул), а відповідь (реакція) створюється відповідно до введених користувачем даних.
- AIML (Artificial Intelligence Markup Language) базується на концепціях теорії розпізнавання образів (Pattern recognition) або концепціях зіставлення зі зразком (Pattern matching).
- Латентно-семантичний аналіз (Latent semantic analysis) можна використовувати разом з AIML для розробки чат-ботів. Він використовується для виявлення подібностей між словами як векторне представлення.
- Chatscript, будучи спадкоємцем мови AIML, є експертною системою, яка складається з мови сценаріїв з відкритим вихідним кодом і механізму, який її запускає. Chatscript складається з правил, які пов'язані з темами, пошуку найкращого елемента, який відповідає рядку запиту користувача, і виконання правила в цій темі.
- RiveScript – це мова рядкового сценарію з простим текстом для розробки чат-ботів та інших розмовних об'єктів. RiveScript має відкритий код із доступними інтерфейсами для Go, Java, JavaScript, Perl і Python.
- Обробка природної мови (NLP) – це область штучного інтелекту, що досліджує комп'ютерне маніпулювання текстів або мовлення природної мови.
- Розуміння природної мови (NLU) є основою будь-якого завдання NLP. Це техніка для реалізації природних інтерфейсів користувача, таких як чат-бот. NLU прагне витягти контекст і значення з введених користувачами природних мов, які можуть

бути неструктурованими та відповідати належним чином відповідно до намірів користувача [4].

Рівень інтелекту бота залежить виключно від того, як він запрограмований. Чат-бот на основі машинного навчання працює краще, адже він розуміє не тільки команди, а й мову. Тому, щоб отримати релевантні відповіді, користувачеві не потрібно вводити точні слова. Крім того, бот вчиться на взаємодіях з клієнтами і може легко вирішувати схожі ситуації, коли вони виникають. Кожен такий діалог покращує інтелект бота [33].

Чат-боти можуть використовувати нейронні мережі як модель мови. Наприклад, генеративні попередньо підготовлені трансформатори (GPT), які використовують трансформаторну архітектуру, стали звичайними для створення складних чат-ботів. «Попереднє навчання» в його назві відноситься до процесу початкового навчання на великому текстовому корпусі, який забезпечує міцну основу для того, щоб модель добре справлялася з подальшими завданнями з обмеженою кількістю даних, пов'язаних із завданням. Прикладом чат-бота GPT є ChatGPT. Незважаючи на критику щодо його точності, ChatGPT привернув увагу своїми детальними відповідями та історичними знаннями. Іншим прикладом є BioGPT, розроблений Microsoft, який фокусується на відповідях на біомедичні запитання.

Створення та впровадження чат-ботів все ще є областю, що розвивається, тісно пов'язаною зі штучним інтелектом і машинним навчанням, тому надані рішення, хоч і мають очевидні переваги, мають деякі важливі обмеження щодо функціональних можливостей і варіантів використання. Однак з часом це змінюється.

Нижче наведено найпоширеніші обмеження [22].

- Оскільки база даних вводу/виводу є фіксованою та обмеженою, чат-боти можуть вийти з ладу під час обробки незбереженого запиту.

- Ефективність чат-бота значною мірою залежить від обробки мови та обмежена через порушення, такі як акценти та орфографічні чи граматичні помилки.
- Чат-боти не можуть відповідати на кілька запитань одночасно, тому можливості спілкування обмежені.
- Для навчання чат-ботам потрібна велика кількість розмовних даних. Генеративні моделі, які базуються на алгоритмах глибокого навчання для створення нових відповідей слово за словом на основі введення користувача, зазвичай навчаються на великому наборі даних фраз природної мови.
- Чат-ботам важко керувати нелінійними бесідами, які повинні відбуватись природньо та відповідати темі розмови з користувачем.
- Як це зазвичай буває з технологічними змінами в існуючих сервісах, деяким споживачам, частіше за все старшого покоління, незручно працювати з чат-ботами через їх обмежене розуміння, що робить очевидним, що їхні запити обробляються машинами.

1. 1. 3. Основи концепції, поняття та алгоритми роботи DNN

Щоб розуміти нейронні мережі краще необхідно для початку розглянути глибоке навчання (deep learning).

У глибокому навчанні кожен рівень вчиться перетворювати свої вхідні дані в дещо більш абстрактне та складне представлення [21]. У програмі розпізнавання зображень необроблений вхід може бути матрицею пікселів; перший репрезентативний шар може абстрагувати пікселі та кодувати краї; другий рівень може складати та кодувати розташування країв; третій шар може кодувати ніс і очі; і четвертий рівень може розпізнати, що зображення містить обличчя. Важливо, що процес глибокого навчання може самостійно дізнатися, які функції оптимально розмістити на якому рівні. Це не позбавляє від необхідності ручного налаштування; наприклад, різна кількість шарів і розміри шарів можуть забезпечити різний ступінь абстракції.

Прикметник «глибокий» у глибокому навчанні відноситься саме до використання кількох рівнів у мережі. Рання робота показала, що лінійний перцептрон не може бути універсальним класифікатором, але мережа з неполіноміальною функцією активації з одним прихованим шаром необмеженої ширини може [21]. Глибоке навчання – це сучасна варіація, яка стосується необмеженої кількості шарів обмеженого розміру, що дозволяє практичне застосування та оптимізоване впровадження, зберігаючи теоретичну універсальність у помірних умовах [22]. У глибокому навчанні також дозволено, щоб рівні були неоднорідними та значно відхилялися від біологічно обґрунтованих коннекціоністських моделей заради ефективності, можливості навчання та розуміння.

Глибоке навчання – це клас алгоритмів машинного навчання, які використовують кілька рівнів для поступового вилучення функцій вищого рівня з не-оброблених вхідних даних. Наприклад, при обробці зображень нижчі рівні можуть ідентифікувати краї, тоді як вищі рівні можуть ідентифікувати поняття, пов'язані з людиною, такі як цифри, літери чи обличчя.

З іншого боку, якщо дивитися на глибоке навчання, глибоке навчання означає «комп'ютерне моделювання» або «автоматизацію» процесів навчання людини від джерела (наприклад, зображення собак) до об'єкта, що вивчається (собак). Тому поняття «глибше» навчання або «найглибше» навчання має сенс. Найглибше навчання відноситься до повністю автоматичного навчання від джерела до кінцевого вивченого об'єкта. Таким чином, глибше навчання відноситься до змішаного процесу навчання: процес навчання людини від джерела до вивченого напівоб'єкта, за яким слід комп'ютерний процес навчання від вивченого людиною напівоб'єкта до кінцевого вивченого об'єкта.

Глибока нейронна мережа (DNN) – це штучна нейронна мережа (ANM) із кількома рівнями між вхідним і вихідним рівнями. Існують різні типи нейронних мереж, але вони завжди складаються з одних і тих самих

компонентів: нейронів, синапсів, ваг, упереджень і функцій. Ці компоненти в цілому функціонують подібно до людського мозку, і їх можна навчити, як і будь-який інший алгоритм ML [36].

Глибокі нейронні мережі, як правило, інтерпретуються в термінах універсальної теореми про наближення або ймовірнісного висновку.

Класична теорема про універсальну апроксимацію стосується здатності прямої нейронної мережі з одним прихованим шаром кінцевого розміру апроксимувати неперервні функції. У 1989 році Джордж Цибенко опублікував перший доказ для сигмоїдних функцій активації, а в 1991 році Курт Горнік узагальнив його для прямої багаторівневої архітектури. Нещодавні дослідження також показали, що універсальне наближення також виконується для необмежених функцій активації, таких як випрямлена лінійна одиниця Куніхіко Фукусіма.

Універсальна теорема про наближення для глибоких нейронних мереж стосується пропускну здатності мереж з обмеженою шириною, але глибина може зростати. Лу та ін. довели, що якщо ширина глибокої нейронної мережі з активацією ReLU строго більша за вхідну розмірність, то мережа може апроксимувати будь-яку інтегровну функцію за Лебегом; якщо ширина менша або дорівнює вхідній розмірності, то глибока нейронна мережа не є універсальним апроксиматором.

Імовірнісна інтерпретація походить із сфери машинного навчання. Він містить висновки, а також концепції оптимізації навчання та тестування, пов'язані з підгонкою та узагальненням відповідно. Більш конкретно, імовірнісна інтерпретація розглядає нелінійність активації як кумулятивну функцію розподілу. Імовірнісна інтерпретація призвела до введення відсіву як регуляризатора в нейронних мережах. Імовірнісна інтерпретація була представлена дослідниками, зокрема Хопфілдом, Уїдроу та Нарендрою, і популяризована в опитуваннях, таких як опитування Бішопа.

DNN можуть моделювати складні нелінійні зв'язки. Архітектури DNN генерують композиційні моделі, де об'єкт виражається як багатошарова

композиція примітивів. Додаткові рівні дозволяють комбінувати функції з нижчих рівнів, потенційно моделюючи складні дані з меншою кількістю одиниць, ніж дрібна мережа з такою ж продуктивністю.

DNN, як правило, є мережами прямого зв'язку, в яких дані переходять із вхідного рівня на вихідний без повернення назад [36]. Спочатку DNN створює кар-ту віртуальних нейронів і призначає зв'язкам між ними випадкові числові значення або «ваги». Вагові коефіцієнти та вхідні дані перемножуються та повертають результат між 0 і 1. Якщо мережа не розпізнає певний шаблон, алгоритм регулює ваги. Таким чином алгоритм може зробити певні параметри більш впливовими, доки не визначить правильну математичну операцію для повної обробки даних.

1.2. Аналіз наявних веб-ресурсів лікарень та поліклінік

Завдяки порівняльному аналізу можна виявити переваги та недоліки вже наявних веб-сайтів клінік та поліклінік. Інформація, отримана в процесі аналізу, дозволить обрати напрям у розробці та внести низку пропозицій щодо покращення веб-сайту клініки. Це дозволить виявити, а згодом і зміцнити слабкі сторони, а також навіть уникнути певних помилок при розробці.

Для подальшого аналізу було обрано наступні три веб-сайти: Поліклініка №1 - КП "Лікарня №1" ЖМР (cm11.zt.ua/poliklinika-1), Медичний центр Полі+клініка (poli-klinika.com.ua) та Сайт клініки «Медібор» (medibor.com.ua). Їх було охарактеризовано та проаналізовано згідно наступних характеристик, враховуючи описані важливі елементи корисного та легкого у використанні сайту клініки раніше:

- 1) Наскільки зручний сайт для навігації користувачам? Наскільки легко записатися на прийом онлайн? Як легко можна знайти необхідно інформацію про клініку?
- 2) Яка інформація надана на сайті? Який її обсяг?
- 3) Як можна дізнатися необхідну інформацію або записатися на прийом не онлайн? Який наявний зворотній зв'язок для користувачів з клінікою?

Веб-сайт Поліклініки №1 досить простий у мінімалістичному стилі (рис. 1.1). Відразу на головній сторінці міститься великий обсяг інформації про клініку, через що досить важко віднайти необхідну інформацію. Також навігаційне меню зверху сторінки незручне для використання та відсутній пошук по сайту, що загалом ускладнює орієнтацію на сайті. На сайті присутня реєстрація та є функція запису на прийом онлайн, але відповідні посилання недостатньо виділені, тому їх нелегко відразу помітити.

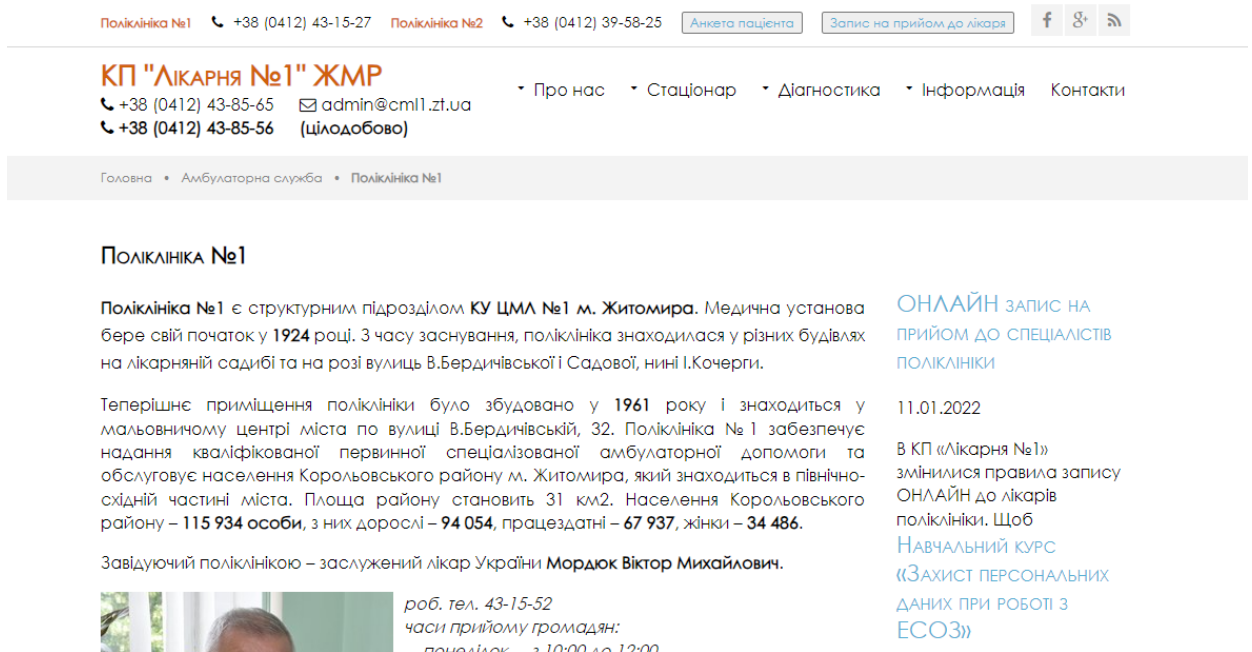


Рис. 1.1. Головна сторінка сайту Поліклініки №1 - КП "Лікарня №1" ЖМР

На сайті присутня інформація про саму поліклініку, її історію заснування та основні дані, а також список лікарів відповідних галузей та список послуг, що надається лікарнею. Також на сайті є блог, але через відсутність пошуку важко орієнтуватись у записах.

Зв'язок з лікарнею можливий або через контактні телефони, або через e-mail. Також надано адресу та посилання на сторінку лікарні у Facebook.

Веб-сайт Полі+клініки також має мінімалістичний дизайн, але більш модерний у порівнянні з попередньо розглянутим сайтом (рис. 1.2). Інформацію викладено стисло та більш детальну інформацію можна знайти на відповідних сторінках. На даному веб-сайті також відсутній пошук, що певним

чином ускладнює навігацію на сайті. Також на сайті відсутня можливість запису онлайн, запис може бути лише через електронну пошту або через відповідні контакти.

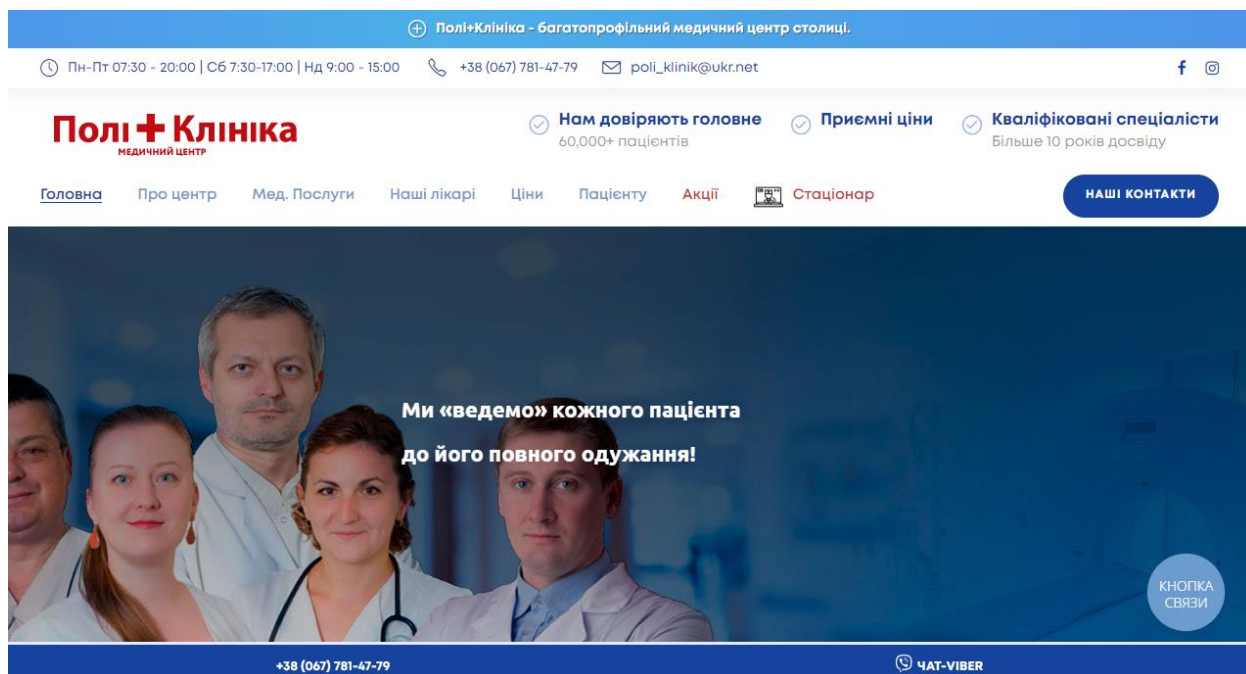


Рис. 1.2. Головна сторінка сайту Полі+клініки

На сайті надано інформацію про лікарню, її створення та устаткування. Надано список лікарів у відповідних галузях, а також список послуг, які надаються, та їхня вартість, а також можливі акції та пільги для користувачів. Також на сайті є посилання до фотогалереї, де можна переглянути, як виглядає клініка та відповідні кабінети і устаткування, що використовуються.

Зв'язок з лікарнею можливий через гарячу лінію або через чат у месенджері Viber, посилання на яке надається на сайті. Також зв'язок можливий через контактні дані, електронну пошту та посилання на сторінки клініки у соціальних мережах.

Як і у попереднього сайту, у веб-сайту Медібор мінімалістичний та сучасний дизайн (рис. 1.3). Інформацію викладено стисло та більш детальну інформацію можна знайти на відповідних сторінках або за допомогою наявного пошуку, що полегшує навігацію на сайті користувачам. Також на сайті наявна можливість запису на прийом онлайн.

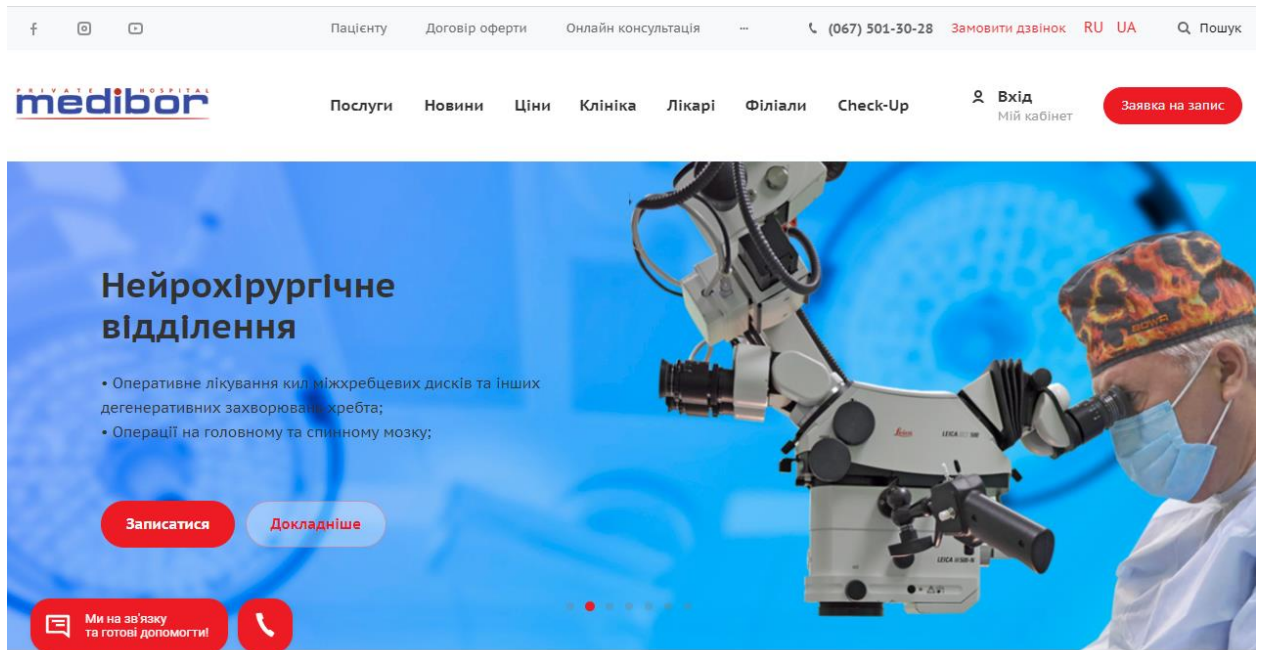


Рис. 1.3. Головна сторінка сайту Медібор

На сайті надано інформацію про лікарню, її створення та устаткування. Надано список лікарів у відповідних галузях, а також список послуг, які надаються, та їхня вартість. Також на сайті є блог про новини та акції, а також посилання для онлайн-консультацій, якщо користувач сайтом не має можливості фізично відвідати клініку.

Зв'язок з лікарнею можливий через гарячу лінію. Також зв'язок можливий через контактні дані, електронну пошту та посилання на сторінки клініки у соціальних мережах. А також на веб-сайті доступний чат-бот, який може надати допомогу користувачам з навігацією або дати відповіді на певні запитання користувачів, що також полегшує користування сайтом та навігацію сторінками.

1.3. Обґрунтування обраних технологій і інструментів

Серверна частина сайту буде розроблена за допомогою Node.js. Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript [11]. В платформі використовується розроблений компанією Google рушій V8. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для

обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

- асинхронна однопоточна модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8;

Для керування модулями використовується пакетний менеджер npm (node package manager).

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується `epoll`, `kqueue`, `/dev/poll` і `select`. Для мультиплексування з'єднань використовується бібліотека `libuv`, для створення пулу нитей (thread pool) задіяна бібліотека `libeio`, для виконання DNS-запитів у неблокуючому режимі інтегрований `c-ares`. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу нитей і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe) [11].

За своєю суттю Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб застосунку, що працює в браузері.

При цьому управління миттєво перейде до коду який слідує після виклику функції `db.query`, а результат запиту буде оброблений як тільки будуть оброблені дані. Жодна функція в Node.js не повинна безпосередньо виконувати (блокуючі) операції вводу/виводу — для отримання даних з диска, від іншого процесу або з мережі потрібна установка callback-обробника.

Для розширення функціональності застосунків на базі Node.js підготовлена велика колекція модулів, в якій можна знайти реалізацію HTTP, SMTP, XMPP, DNS, FTP, IMAP, POP3 серверів і клієнтів, модулі для інтеграції з різними вебфреймворками, обробники WebSocket і AJAX, конектори до СКБД (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори, CSS-пуші, реалізації криптоалгоритмів і систем авторизації (наприклад, OAuth), XML-парсери.

Стандартна поставка node.js включає в себе кількадесят модулів, у яких реалізовані типові операції для взаємодії з операційною системою, файловою системою, мережею і протоколами, утиліти для обробки даних.

Крім того, є доступними безліч модулів від незалежних розробників, програмісти можуть отримати їх з відкритих репозиторіїв і використовувати у своїх проектах.

Клієнтська частина буде розроблена за допомогою React. React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків [24].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим [9]. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків.

Коли справа доходить до створення чат-ботів, існує кілька доступних фреймворків і бібліотек. Однак React.js став одним із найпопулярніших варіантів для розробки чат-ботів.

Перевагами використання React.js для розробки чат-ботів є наступне [24]:

1. Інтеграція з бібліотеками NLP:

NLP – це напрямок штучного інтелекту, який зосереджується на взаємодії між людьми та машинами за допомогою природної мови. Інтегруючи React.js із потужними бібліотеками NLP, такими як TensorFlow.js або Natural, розробники можуть покращити здатність своїх чат-ботів ефективніше розуміти запити користувачів і відповідати на них.

Бібліотеки NLP забезпечують широкий спектр функціональних можливостей, таких як аналіз настроїв, розпізнавання іменованих сутностей і класифікація намірів. Ці можливості дозволяють чат-ботам інтерпретувати та аналізувати введені користувачем дані, що веде до більш точних і відповідних контексту відповідей. За допомогою React.js розробники можуть легко використовувати потужність бібліотек NLP, що робить його ідеальним вибором для створення інтелектуальних чат-ботів.

2. Багатомовна підтримка

Чат-боти все частіше використовуються в усьому світі, і обслуговування користувачів, які розмовляють різними мовами, має вирішальне значення для їх успіху. За допомогою React.js розробники можуть легко працювати з кількома мовами та гарантувати, що їхні чат-боти можуть ефективно спілкуватися з користувачами з різним мовним походженням.

React.js забезпечує вбудовану підтримку інтернаціоналізації (i18n) через такі бібліотеки, як React-Intl, що дозволяє легко перекладати та локалізувати інтерфейси чат-ботів. Розробники можуть створювати компоненти для певної мови та ефективно керувати перекладами, дозволяючи чат-ботам взаємодіяти з користувачами мовою, яку вони віддають перевагу. Ця багатомовна підтримка надає чат-ботам ширший охоплення та дозволяє компаніям ефективніше взаємодіяти з глобальною аудиторією.

3. Послідовний інтерфейс користувача

Підтримка узгодженого інтерфейсу користувача (UI) є важливою для будь-якої програми, і чат-боти не є винятком. React.js відмінно підходить для створення повторно використовуваних компонентів інтерфейсу користувача, дозволяючи розробникам підтримувати узгоджений вигляд і відчуття на різних екранах чат-ботів. Ця узгодженість покращує взаємодію з користувачем і робить чат-бот схожим на бездоганне розширення бренду чи послуги, яку він представляє.

React.js дотримується компонентної архітектури, де кожен елемент інтерфейсу інкапсульовано в самодостатній компонент. Ці компоненти можна легко повторно використовувати та комбінувати для створення складних інтерфейсів чат-бота. Крім того, React.js надає віртуальну DOM (об'єктну модель документа), яка ефективно оновлює лише необхідні компоненти, коли є зміни, що призводить до кращої продуктивності та швидшого відтворення.

4. Оновлення в реальному часі та швидкість реагування:

React.js перевершує цю сферу завдяки своїй віртуальній DOM (Document Object Model), яка дозволяє ефективно та швидко оновлювати без перезавантаження всієї веб-сторінки. Це означає, що відповіді чат-бота можна оновлювати миттєво, забезпечуючи інтерактивний і чутливий інтерфейс користувача.

Крім того, React.js використовує компонентну архітектуру, що ще більше покращує його здатність обробляти оновлення в реальному часі. Компоненти можна легко повторно використовувати та оновлювати незалежно, що полегшує керування та підтримку інтерфейсу чат-бота з часом. За допомогою React.js розробники можуть гарантувати, що їхні чат-боти завжди оновлені та швидко реагують, забезпечуючи користувачам безперебійну та привабливу роботу.

5. Керування станом:

React.js забезпечує чудову систему керування станом завдяки використанню атрибутів і стану. Реквізити та стан дозволяють розробникам

ефективно відстежувати та керувати даними чат-бота та взаємодією користувачів.

За допомогою React.js розробники можуть легко оновлювати стан чат-бота на основі даних користувача та інших зовнішніх факторів. Це дає змогу створювати динамічні чат-боти, які адаптуються до потреб користувача та надають персоналізовані відповіді. Здатність ефективно керувати станом гарантує, що чат-бот може обробляти складні розмови та надавати точну та актуальну інформацію.

6. Інтеграція із зовнішніми джерелами:

Інтеграція чат-бота із зовнішніми джерелами, такими як API або бази даних, часто необхідна для надання користувачам актуальної інформації. React.js спрощує цей процес, пропонуючи повну інтеграцію із зовнішніми джерелами. Його гнучка архітектура дозволяє розробникам отримувати та оновлювати дані з різних джерел, не порушуючи функціональність чат-бота.

React.js також отримує переваги від великої екосистеми бібліотек та інструментів, які полегшують інтеграцію із зовнішніми джерелами. Незалежно від того, чи потрібно вам отримати дані з REST API чи оновити базу даних, React.js надає необхідні інструменти та ресурси для оптимізації процесу. Це гарантує, що ваш чат-бот може надавати користувачам точну та актуальну інформацію, підвищуючи його загальну функціональність і ефективність.

7. Спеціальні можливості

Доступність — важливий аспект будь-якого чат-бота. React.js надає широкий спектр спеціальних можливостей, які гарантують, що вашим чат-ботом зможуть користуватися всі, включно з людьми з обмеженими можливостями. Це включає підтримку атрибутів ARIA (Accessible Rich Internet Applications), які дозволяють додавати семантичну інформацію до елементів вашого чат-бота, роблячи їх більш доступними для програм зчитування з екрана та допоміжних технологій. Крім того, React.js забезпечує легку інтеграцію з клавіатурною навігацією, гарантуючи, що користувачі

можуть взаємодіяти з вашим чат-ботом, використовуючи лише свою клавіатуру.

8. Контекстна інформація

Чат-ботам часто потрібно відображати користувачам релевантну інформацію. React.js дозволяє легко керувати цією інформацією та відображати її завдяки своїй компонентній архітектурі. Ви можете створювати повторно використовувані компоненти, які інкапсулюють певну функціональність і легко вводять у них необхідні контекстні дані. Це дозволяє створювати чат-ботів, які можуть динамічно адаптуватися до введених користувачами та відображати відповідну інформацію, створюючи більш персоналізований та привабливий досвід користувача.

9. Тестування та налагодження

Тестування та налагодження необхідні для забезпечення надійності та продуктивності вашого чат-бота. React.js забезпечує надійне середовище тестування та інструменти налагодження, які спрощують процес. За допомогою таких інструментів, як Jest і Enzyme, ви можете писати модульні тести для компонентів вашого чат-бота та легко симулювати взаємодію користувача. React DevTools, розширення для веб-переглядача, дозволяє перевіряти та налагоджувати ієрархію компонентів, стан і атрибути вашого чат-бота, полегшуючи виявлення та виправлення будь-яких проблем.

Під час створення чат-бота за допомогою React.js доступно кілька бібліотек і інструментів, які можуть покращити процес розробки та надати додаткові функції. Ось кілька популярних бібліотек React.js, спеціально розроблених для розробки чат-ботів. Ці бібліотеки можуть значно пришвидшити процес розробки чат-бота React.js і надати основні функції та компоненти інтерфейсу користувача для створення інтерактивних і привабливих інтерфейсів чат-бота в React.js.

Але для дослідження та розробки було обрано використання мови програмування Python.

Python, безумовно, є найпопулярнішою мовою програмування серед спеціалістів із обробки даних. Дослідження показують, що понад 57% дослідників даних використовували python для машинного навчання.

Python пропонує універсальність. Це проста мова з її можливостями, подібними до людської, завдяки рокам досліджень і розробок, а також узгодженим синтаксисом. Заслуга в цьому належить включенню AIML (мова розмітки штучного інтелекту), яка полегшує розробникам написання синтаксису.

Python також використовується для створення розмовних компонентів, розміщених на веб-сервері, і підтримує велику бібліотеку для чат-ботів AI. Це дає змогу розробникам створювати на Python чат-ботів, які мають розмовні нюанси та емпатичні тони.

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, аспектно-орієнтована та функціональна.

Замість того, щоб будувати всю свою функціональність з нуля, Python був розроблений так, щоб бути дуже розширеним за допомогою модулів. Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих додатків. Бачення Ван Россума про невелику основну мову з великою стандартною бібліотекою та легко розширюваним перекладачем впливає з його розчарувань з ABC, що підтримує протилежний підхід.

Python має бути легко читабельною мовою. Його форматування візуально розбивають і часто використовують англійські ключові слова, де інші мови використовують пунктуацію. На відміну від багатьох інших мов, він не використовує кучеряві кронштейни для розмежування блоків, а крапки з комочками після дозволу заяв, але рідко використовуються. Він має менше синтаксичних винятків та спеціальних випадків, ніж C або Pascal.

ChatterBot – це бібліотека Python, яка дозволяє легко генерувати автоматичні відповіді на введення користувача. ChatterBot використовує набір алгоритмів машинного навчання для створення різних типів відповідей, що полегшує розробникам створення чат-ботів і автоматизацію розмов з користувачами.

Висновки до розділу 1

Під час роботи над першим розділом було проведено науковий аналіз та обґрунтуванні вибору моделей, методів та технічних рішень.

В межах дослідження було розглянуто концепції, методи та алгоритми розробки веб-сайтів для клінік, а також для розробки чат-ботів. Було також проаналізовано наявні прикладі веб-ресурсів для лікарень та клінік. Після чого було визначено завдання, а надалі було розглянуто та обґрунтовано обрані технології і інструменти.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТА КОНЦЕПЦІЙ РОЗРОБКИ ЧАТ-БОТУ

2.1. Розробка чат-боту без використання фрейму ChatterBot

Перш ніж працювати над чат-ботом для готового веб-ресурсу, потрібно спершу зрозуміти основні алгоритми та концепції розробки технології чат-боту за допомогою мови програмування Python.

2.1.1. Розробка чат-боту на основі DNN

Було вирішено, що чат-бот буде працювати на основі DNN (Deep Neural Network), щоб ідентифікувати шаблони речень, наданих користувачем як вхідні дані, і обирати випадкову відповідь, пов'язану з цим запитом. Чат-бот буде реалізовано за допомогою бібліотек Python, як-от, наприклад, бібліотеки NLTK у Python, що має функції, які допомагають з'ясувати найбільш релевантні слова з речення чи абзацу та зв'язати слова з їх кореневим значенням і можуть скоротити їх. Цей процес відомий як Stemming. Потім слова перетворюються на відповідні числові значення, оскільки нейронні мережі розуміють лише числа. Процес перетворення тексту в числові значення відомий як One-Hot Encoding. Коли попередня обробка даних буде завершена, ми створимо нейронні мережі за допомогою «TFlearn», а потім підберемо в них навчальні дані. Після успішного навчання модель повинна бути здатна передбачати теги, пов'язані із запитом користувача.

Для навчання нашого чат-боту нам знадобиться файл «intents.json», лістинг якого представлено на рис. 2.1-2.2.

Кожен намір (intent) містить тег, шаблони, відповіді та контекст. Шаблони – це дані, які користувач, швидше за все, введе, а відповіді – це результати чат-бота. Цей файл містить загальні фрази привітань, а також запитання про те, які лікарі працюють в клініці та запитання про прийом, як-от їхня вартість, в які дні можна записуватись, а також як можна записатися на прийом.

Тепер ми можемо завантажити файл «intents.json» і почати процес.

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
      "responses": ["Hello, how can I help?", "Hi there, how can I help?", "Hello, do you have any questions?"],
      "context": [""]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
      "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
      "context": [""]
    },
    {
      "tag": "thanks",
      "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
      "responses": ["My pleasure", "You're welcome"],
      "context": [""]
    },
    {
      "tag": "visitTime",
      "patterns": ["When can I sign up for a checkup?", "Time of the checkups", "When can I come for a visit", "Time of the visits", "Time"],
      "responses": ["Any day of the week between 9am and 8pm"],
      "context": [""]
    },
    {
      "tag": "doctorSpeciality",
      "patterns": ["What doctors are there?", "What specialists are there?", "What kind of doctors are there?", "List of specialties", "Doctors"],
      "responses": ["There are specialists in gastroenterology, cardiology, orthopedy, otolaryngology and more. Please refer to our specialists page to"],
      "context": [""]
    }
  ],
}
```

Рис. 2.1. Лістинг файлу «intents.json»

```
{
  "tag": "prices",
  "patterns": ["How much does it cost?", "How much it costs?", "How much for a visit?", "What are the prices?", "Prices", "Prices"],
  "responses": ["100 for a consultation, 120 for a check-up and 200 for an analysis"],
  "context": [""]
},
{
  "tag": "signup",
  "patterns": ["How do I sign up?", "How to sign up for a visit?", "How to make a visit?", "I want to sign for a visit"],
  "responses": ["Please, open the form for a sign-up and enter your info, pick the specialist you would like to visit and comfortable date and time"],
  "context": [""]
},
{
  "tag": "help",
  "patterns": ["Hotline phone", "Do you have a hotline phone number?", "Do you have a hotline number?", "I want to sign up by phone"],
  "responses": ["Of course, please dial +380961012443", "It is +380961012443", "You can contact hotline by calling +380961012443"],
  "context": [""]
}
}
```

Рис. 2.2. Лістинг файлу «intents.json»

Тепер завдання полягає в тому, щоб змусити модель вивчити співвідношення між шаблонами та тегамі, щоб, коли користувач вводить певну фразу, він міг ідентифікувати відповідний тег і видати одну з відповідей як результат запиту.

```

with open('intents.json') as intents:
    data = json.load(intents)

stemmer = LancasterStemmer()

words = []
labels = []
x_docs = []
y_docs = []

for intent in data['intents']:
    for pattern in intent['patterns']:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        x_docs.append(wrds)
        y_docs.append(intent['tag'])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])

```

Рис. 2.3. Лістинг завантаження файлу «intents.json»

Найпершим чином необхідно завантажити модель «Punkt», що надається бібліотекою NLTK (рис. 2.4).

```

import nltk

nltk.download("punkt")

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True

```

Рис. 2.4. Завантаження моделі «Punkt»

У NLTK PUNKT є моделлю, що навчається без нагляду, що означає, що її можна навчати на немаркованих даних (дані, до яких не було додано інформацію про характеристики, властивості чи категорії, називаються немаркованими даними).

Вона генерує список речень із тексту, розробляючи модель для слів, які починають речення, прийменникові фрази та скорочення за допомогою техніки без нагляду. Без попереднього використання, її потрібно навчити на значній кількості відкритого тексту призначеною мовою.

Далі розглянемо бібліотеки та модулі, які будуть використовуватись для реалізації чат-боту. Список імпортованих бібліотек представлено на рис. 2.5.

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
import numpy as np
import tflearn
import tensorflow as tf
import json
import pickle
import random
```

Рис. 2.5. Імпорт модулів

Набір інструментів природної мови, або частіше NLTK, – це набір бібліотек і програм для символної та статистичної обробки природної мови (NLP) для англійської мови, написаних мовою програмування Python. Його розробили Стівен Берд і Едвард Лопер з кафедри комп'ютерних та інформаційних наук університету Пенсільванії. NLTK містить як набори даних, так і графічні матеріали.

NLTK призначений для підтримки досліджень і викладання навчальних курсів пов'язаних з НЛП та близькоспорідненими областями, включаючи емпіричну лінгвістику, когнітивну науку, штучний інтелект, пошук інформації та машинне навчання. NLTK успішно використовується як навчальний інструмент, а також як платформа для створення прототипів і побудови дослідницьких систем.

Під час роботи над будь-яким проектом у сфері обробки природної мови `nltk` є найважливішим модулем, який використовується.

TensorFlow – відкрита програмна бібліотека для машинного навчання цілої низки задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди. Її наразі застосовують як для досліджень, так і для розробки продуктів Google.

TensorFlow надає бібліотеку готових алгоритмів чисельних обчислень, реалізованих через графи потоків даних (data flow graphs). Вузли в таких

графах реалізують математичні операції або точки входу/виводу, в той час як ребра графу представляють багатовимірні масиви даних (тензори), які перетікають між вузлами. Вузли можуть бути закріплені за обчислювальними пристроями і виконуватися асинхронно, паралельно обробляючи разом все підходящі до них тензори, що дозволяє організувати одночасну роботу вузлів в нейронній мережі за аналогією з одночасною активацією нейронів в мозку.

Однак TensorFlow є бібліотекою низького рівня. Використовувати TensorFlow безпосередньо досить складно, оскільки API надзвичайно багатослівний і схильний до тонких дій. У TensorFlow важко виявити помилки. Щоб подолати ці труднощі, була розроблена оболонка під назвою TFLearn.

TFLearn – це модульна та прозора бібліотека глибокого навчання, створена на основі Tensorflow. Вона була розроблена, щоб забезпечити API вищого рівня для бібліотеки TensorFlow, щоб полегшити та прискорити експерименти, залишаючись при цьому повністю прозорою і сумісною з нею.

Після попереднього завантаження файлу «intents.json» і отримання даних необхідно почати попередню обробку даних, як показано на рис. 2.6.

```
words = [stemmer.stem(w.lower()) for w in words if w not in "?"]
words = sorted(list(set(words)))
labels = sorted(labels)
```

Рис. 2.6. Попередня обробка даних

Відбувається розпізнавання слів, а також видалення повторюваних слів зі списку слів. Тепер використаємо алгоритм Lancaster Stemmer, щоб скоротити слова до їхньої основи (рис. 2.7).

Спочатку коротко розглянемо, що таке стемінг.

Стеммінг – це процес скорочення слова до його основи. Основа слова – це основа або коренева форма слова, і не обов’язково бути існуючим словом.

Ланкастер є одним із найагресивніших стеммерів, оскільки він має тенденцію перевищувати багато слів. Стеммер Ланкастера складається з набору правил, де кожне правило визначає або видалення, або заміну

закінчення. Крім того, деякі правила обмежені неушкодженими словами, а деякі правила застосовуються ітеративно, коли слово проходить через них.

Через суворі правила існують дві додаткові умови, щоб запобігти походженню різних короткокореневих слів:

- Якщо слово починається з голосної, то принаймні дві літери повинні залишитися після кореня (owing -> ow, але не ear -> e).

- Якщо слово починається з приголосної, то принаймні три літери повинні залишитися після основної букви, і принаймні одна з них має бути голосною або «у» (говоріння -> say, але не string -> str).

Заголом стеммер Lancaster має понад 100 правил.

```
training = []
output = []
out_empty = [0 for _ in range(len(labels))]

for x, doc in enumerate(x_docs):
    bag = []
    wrds = [stemmer.stem(w) for w in doc]
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)

    output_row = out_empty[:]
    output_row[labels.index(y_docs[x])] = 1

    training.append(bag)
    output.append(output_row)

training = np.array(training)
output = np.array(output)
```

Рис. 2.7. One-Hot кодування та підготовка навчальних даних

Отримані тренувальні та вихідні дані закодовані в режимі One-Hot. Буквально, слова перетворюються у форму одиниць і нулів, які потім додаються до навчального списку, а також вихідного списку, а потім перетворюються на масиви NumPy.

Тепер нам потрібно створити нашу модель за допомогою нейронних мереж (рис. 2.8).

```

net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, len(output[0]), activation='softmax')
net = tflearn.regression(net)

model = tflearn.DNN(net)
model.fit(training, output, n_epoch=500, batch_size=8, show_metric=True)

```

Рис. 2.8. Створення моделі за допомогою нейронних мереж

Перший шар є вхідним шаром з параметром рівновеликих вхідних даних. Далі три середні – це приховані рівні, які відповідають за всю обробку вхідних даних. Вихідний рівень надає ймовірності різних слів у навчальних даних.

Дані навчання вписуються в модель і встановлюються епохи на 500, у яких навчання триватиме, доки не досягне 500 ітерацій. Епоха в нейронній мережі – це навчання нейронної мережі з усіма навчальними даними за один цикл. Використання більше, ніж однієї епохи, необхідно, щоб отримати хорошу ефективність ненавчальних даних (на практиці це можна приблизно оцінити за допомогою набору затримок); зазвичай (але не завжди) це займає більше ніж один прохід тренувальних даних.

Ми навчили модель зі списком слів або так званим мішком слів, тому, щоб робити прогнози, нам також потрібно робити те саме. Тепер ми можемо створити функцію, яка надасть нам пакет слів для прогнозування нашої моделі (рис. 2.9).

```

def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]
    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for s_word in s_words:
        for i, w in enumerate(words):
            if w == s_word:
                bag[i] = 1
    return np.array(bag)

```

Рис. 2.9. Функцію для надання пакету слів для прогнозування моделі

Ця функція допомагає створити пакет слів для нашої моделі.

Тепер необхідно створити функцію чату, яка об'єднає все це разом (рис. 2.10).

```
def chat():

    while True:
        inp = input("\n\nYou: ")
        if inp.lower() == 'quit':
            break
        results = model.predict([bag_of_words(inp, words)])

        results_index = np.argmax(results)

        tag = labels[results_index]

        for tg in data['intents']:

            if tg['tag'] == tag:
                responses = tg['responses']
                print("Bot:\t" + random.choice(responses))
```

Рис. 2.10. Функція чату

Спочатку модель передбачає результати, використовуючи пакет слів і дані користувача, а потім повертає список ймовірностей. Серед ймовірностей найвище число, швидше за все, буде результатом, який очікує користувач. Отже, ми вибираємо індекс найвищої ймовірності та знаходимо тег і відповіді цього конкретного індексу. Тоді ми можемо вибрати деякі випадкові відповіді зі списку відповідей.

На рис. 2.11 представлено, як чат-бот може фактично ідентифікувати шаблон введених фраз користувача та відповідати відповідно.

```
Training Step: 2999 | total loss: 0.00517 | time: 0.040s
| Adam | epoch: 500 | loss: 0.00517 - acc: 1.0000 -- iter: 40/41
Training Step: 3000 | total loss: 0.00550 | time: 0.046s
| Adam | epoch: 500 | loss: 0.00550 - acc: 1.0000 -- iter: 41/41
--

You: Hi
Bot: Hi there, how can I help?

You: List of specialties
Bot: There are specialists in gastroenterology, cardiology, orthopedy, otolaryngology and more. Please refer to our specialists page to learn more

You: How much it costs?
Bot: 100 for a consultation, 120 for a check-up and 200 for an analysis

You: Thanks
Bot: My pleasure

You: quit
```

Рис. 2.11. Приклад роботи чат-боту

Видно, що після проведеного навчання чат-бот дає вірні відповіді на запити від користувача. У результаті введення звичайних фраз чат-бот дає відповідь на запитання, а у результаті введення quit – припиняється спілкування.

2.1.2. Аналіз роботи чат-боту та його результати в діаграмах і графіках

Для візуального аналізу чат-боту було застосовано TensorBoard. TensorBoard – це інструмент для забезпечення вимірювань і візуалізацій, необхідних під час робочого процесу машинного навчання. Це дає змогу відстежувати такі показники експерименту, як втрати та точність, візуалізувати графік моделі, проектувати вбудовування в нижчий вимірний простір і багато іншого.

Спочатку необхідно оновити код наступним чином, як показано на рис 2.12.

```
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
model = tflearn.DNN(net, tensorboard_verbose=3, tensorboard_dir=log_dir)
```

Рис. 2.12. Лістинг змін у коді

TFLearn підтримує детальний рівень для автоматичного керування підсумками:

- 0: втрати та показники (найкраща швидкість).
- 1: Втрата, метрика та градієнти.
- 2: втрати, метрика, градієнти та ваги.
- 3: втрати, показники, градієнти, ваги, активації та розрідженість (найкраща візуалізація).

Для властивості `tensorboard_verbose` було обрано значення 3 для найкращої візуалізації.

Відповідно логи будуть зберігатися в підкаталогах з міткою часу, щоб можна було легко вибрати різні тренувальні прогони. Для цього створено змінну `log_dir`, яка зберігає назву підкаталогу.

Потім можна запустити Tensorboard для візуалізації продуктивності.

Інформаційна панель TensorBoard Graphs – це інструмент для вивчення вашої моделі TensorFlow. Можна швидко переглянути концептуальний графік структури моделі та переконатися, що вона відповідає запланованому дизайну. Також можна переглянути графік операційного рівня, щоб зрозуміти, як TensorFlow розуміє програму. Вивчення графіка операційного рівня може дати вам уявлення про те, як змінити свою модель. Саме цей інструмент буде використовуватися для візуального аналізу.

У результаті отримуємо наступні графіки точності та втрат під час навчання, які представлено на рис. 2.13-2.14.

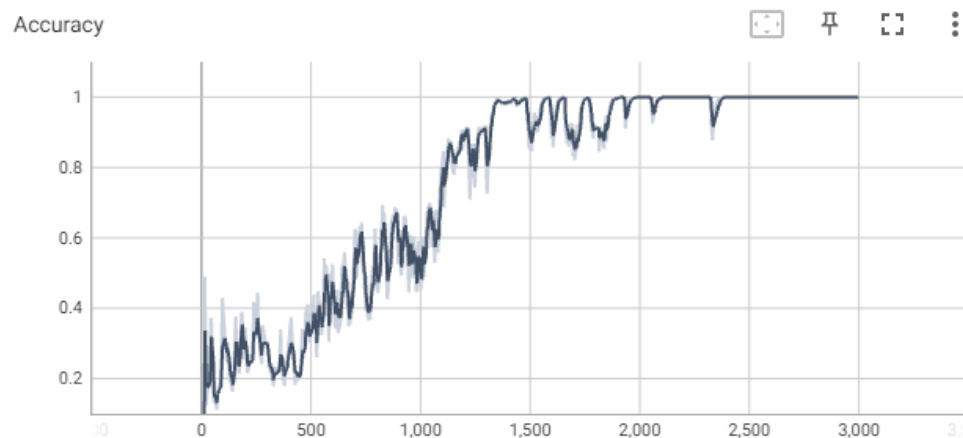


Рис. 2.13. Графік точності

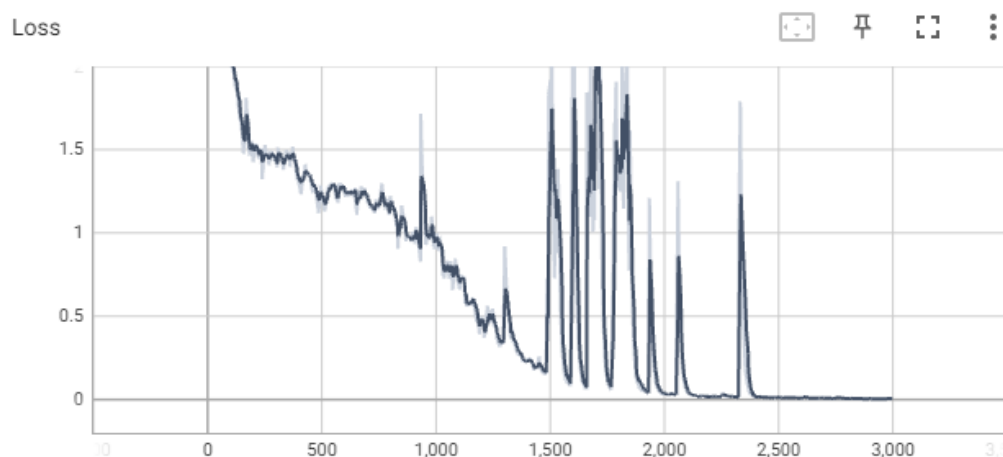


Рис. 2.14. Графік втрат під час навчання

У більшості випадків спостерігається, що точність зростає зі зменшенням втрат, але це не завжди так. Точність і втрати мають різні

визначення та вимірюють різні речі. Вони часто здаються обернено пропорційними, але математичної залежності між цими двома показниками немає.

Мати низьку точність, але високі втрати означає, що модель робить великі помилки в більшості даних. Але якщо і втрати, і точність низькі, це означає, що модель робить невеликі помилки в більшості даних. Однак якщо обидва показники високі, це призводить до великих помилок у деяких даних. Нарешті, якщо точність висока, а втрати низькі, тоді модель допускає невеликі помилки лише в деяких даних, що було б ідеальним випадком.

На основі отриманих даних можна зробити висновок, що чат-бот швидко вчиться на основі наданих йому даних і робить невелику кількість помилок, що також свідчить про невисоку складність наданих даних.

2.2. Розробка чат-боту за допомогою фрейму ChatterBot

Ненавчений екземпляр ChatterBot починає без знання того, як спілкуватися. Кожного разу, коли користувач вводить заяву, бібліотека зберігає текст, який він ввів, і текст, на який заява була відповіддю. Оскільки ChatterBot отримує більше даних, збільшується кількість відповідей, на які він може відповісти, і точність кожної відповіді щодо вхідного оператора.

Програма вибирає найбільш відповідну відповідь, шукаючи найближче відповідне відоме твердження, яке відповідає введеним даним, а потім вибирає відповідь із списку відомих відповідей на це твердження (рис. 3.15).

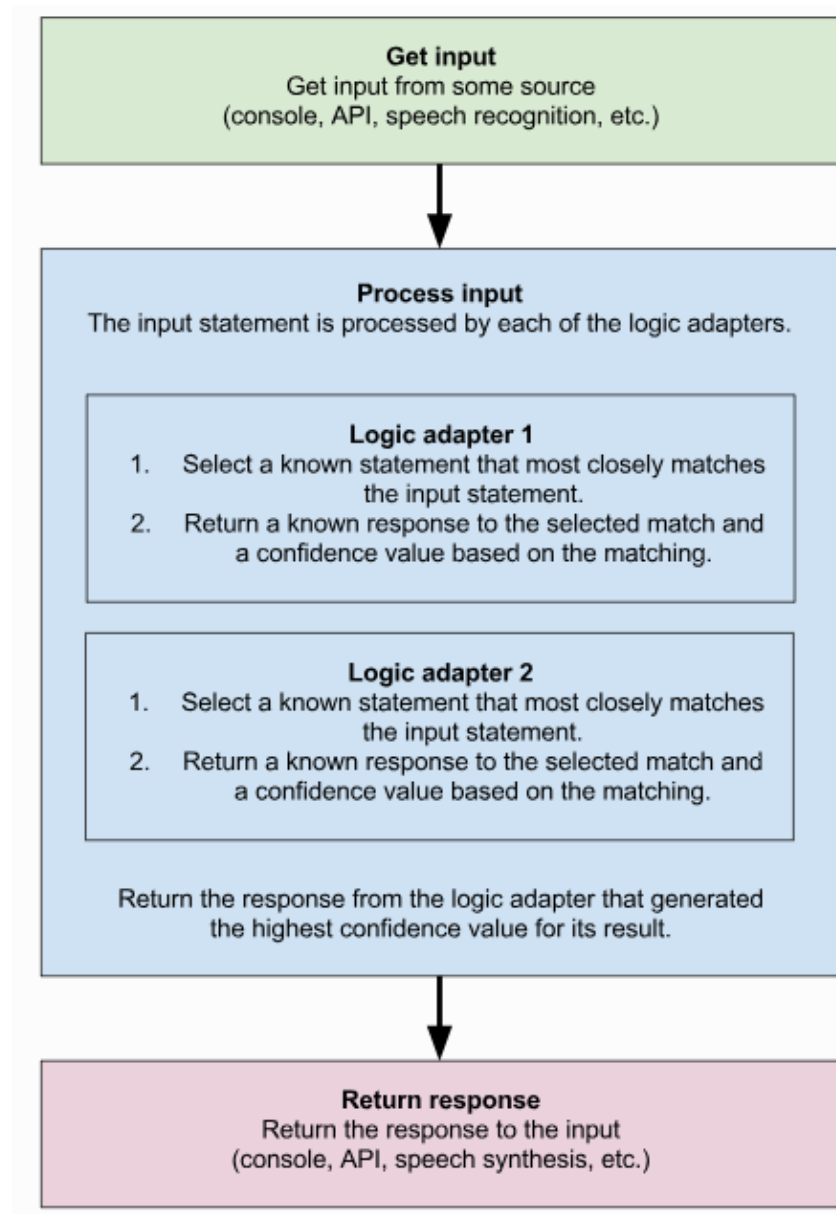


Рис. 3. 15. Схема процесу роботи ChatterBot

Враховуючи процес роботи фрейму для початку ініціалізуємо екземпляр ChatBot як показано на рис. 3.16.

```

bot = ChatBot('MedBot', read_only=True,
              preprocessors=[ 'chatterbot.preprocessors.convert_to_ascii',
                              'chatterbot.preprocessors.unescape_html',
                              'chatterbot.preprocessors.clean_whitespace'],
              logic_adapters=[
                  {
                      'import_path': 'chatterbot.logic.BestMatch',
                      'default_response': 'Sorry, I am unable to process your request.',
                      'maximum_similarity_threshold': 0.90
                  }
              ],)
  
```

Рис. 3.16. Ініціалізація екземпляра ChatBot

Препроцесори ChatterBot – це прості функції, які змінюють вхідний оператор, який отримує чат-бот перед обробкою оператора логічним адаптером.

`chatterbot.preprocessors.convert_to_ascii`: Перетворює символи Unicode на еквіваленти символів ASCII. Наприклад: «på fédéral» стає «pa federal».

`chatterbot.preprocessors.unescape_html`: Перетворює екрановані символи html на неекрановані символи html. Наприклад: «» стає «'».

`chatterbot.preprocessors.clean_whitespace`: Видаляє будь-які послідовні пробіли з вхідного запиту.

Логічні адаптери визначають логіку того, як ChatterBot вибирає відповідь на заданий вхідний оператор.

Типовий логічний адаптер, призначений для повернення відповіді на вхідний оператор, буде використовувати для цього два основні кроки. Перший крок передбачає пошук у базі даних відомого твердження, яке відповідає або близько відповідає вхідному твердженню. Після вибору збігу другий крок передбачає вибір відомої відповіді на вибраний збіг. Часто буде кілька існуючих тверджень, які є відповідями на відомий збіг, та логічний адаптер `BestMatch` вибирає відповідь на основі найбільш відомого збігу з даним оператором.

Адаптер найкращого збігу використовує функцію для порівняння вхідного оператора з відомими операторами. Коли він знаходить найбільш близьку відповідність до вхідного оператора, він використовує іншу функцію, щоб вибрати одну з відомих відповідей на цей оператор.

`default_response` (str або list або tuple): Відповідь за замовчуванням, яку повертає цей логічний адаптер, якщо немає іншої відповіді для повернення.

`maximum_similarity_threshold`: Максимальна кількість подібності між двома операторами, необхідна для припинення процесу пошуку. Пошук відповідного твердження триватиме, доки не буде знайдено твердження з більшою або рівною подібністю або пошуковий набір не буде вичерпано. За замовчуванням має значення 0,95.

Щоб надати боту певні знання, нам потрібно навчити бота навчальними даними. навчальні дані заповнюються у списку, який представлятиме розмову. ChatterBot використовує обробку природної мови, щоб вивчати дані, надані для бота. На рис. 3.17 надано лістинг файлу з навчальними даними з навігацією сайтом та на рис. 3.18 надано лістинг файлу з навчальними даними з надання базових медичних порад у випадку невеликих медичних проблем.

```

1 Hi
2 Hello, how may I help?
3 Hello
4 Hello, how may I help?
5 Good morning
6 Good morning, how may I help?
7 Good evening
8 Good evening, how may I help?
9 I would like to book an appointment
10 Sure, please click on the Book Appointment button on the Home page.
11 I want to book an appointment
12 Sure, please click on the Book Appointment button on the Home page.
13 I want to see a doctor
14 Sure, please click on the Book Appointment button on the Home page.
15 How to book an appointment?
16 Please click on the Book Appointment button on the Home page.
17 Do I have to book an appointment online?
18 You can call the hospital. Our phone number is (093)-123-34-43.

```

Рис. 3.17. Лістинг файлу з навчальними даними з навігацією сайтом

```

1 What to do if cuts?
2 Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blea
3 How to cure cuts?
4 Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blea
5 Which medicine to apply for cuts?
6 Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blea
7 What to apply on cuts?
8 Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blea
9 Cuts
10 Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blea
11 How do you treat abrasions?
12 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
13 Do abrasions cause scars?
14 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
15 What to do if abrasions?
16 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
17 Which medicine to apply for abrasions?
18 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
19 How to cure abrasions?
20 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
21 Abrasions
22 Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p

```

Рис. 3. 18. Лістинг файлу з навчальними даним з надання базових медичних порад

Далі ці дані необхідно опрацювати, як представлено на рис. 3.19.

```
data1 = open('training_data.txt').read()
conversations = data1.strip().split('\n')

data2 = open('training_help.txt').read()
helping = data2.strip().split('\n')
```

Рис. 3.19. Обробка даних

Після чого оброблені дані ми використовуємо для тренування боту, як представлено на рис. 3.20.

```
trainer.train(conversations)
trainer.train(helping)
```

Рис. 3.20. Тренування чат-боту обробленими даними

Тепер чат-бот готовий до використання. На рис. 3.21 представлено приклад роботи чат-боту.

```
(venv7) E:\vazhno\master\praktika23\client\api>py medchatbot.py
List Trainer: [#####] 100%
Welcome to the ChatBot Service for Med! How can I help you? (Enter 'Bye' to exit)
You:Hello
ChatBot: Hello, how may I help?
You:I want to book an appointment
ChatBot: Sure, please click on the Book Appointment button on the Home page.
You:Can i come without booking
ChatBot: You can visit the hospital without an appointment. If there will be an empty slot a doctor will see you.
You:What services do you offer
ChatBot: We offer check-ups, consultations as well as medical tests.
You:How much is a check-up
ChatBot: The check-up is 100.
You:Which doctors are there
ChatBot: We have specialists in the gastroenterology, cardiology, orthopedy, otolaryngology and endocrinology fields.
You:thanks
ChatBot: Of course!
You:bye
ChatBot: It was great talking to you!
```

Рис. 3.21. Приклад роботи чат-боту

Висновки до розділу 2

В ході виконання другого розділу було розроблено чат-боту без використання фрейму ChatterBot та наведено повний опис джерела даних, що використовується для навчання чат-боту, і описано його структуру, а також представлено і описано реалізовані методи, алгоритми, використані бібліотеки для розробки та навчання чат-боту. Також було проведено візуальний аналіз, описано його результати та за отриманими результатами зроблено висновки про роботу чат-боту. Після чого було розроблено чат-бот за допомогою фрейму ChatterBot, який і буде використовуватись для готового веб-ресурсу, описано реалізовані методи та процес розробки і навчання чат-боту, та представлено приклад його роботи.

РОЗДІЛ 3. РОЗГОРТАННЯ ВЕБ-РЕСУРСУ ТА АНАЛІЗ НЕОБХІДНОСТІ ВИКОРИСТАННЯ ЧАТ-БОТІВ

3.1. Розгортання веб-ресурсу

Як було зазначено в розділі 1 клієнтська частина сайту була розроблена за допомогою використання React. Головна сторінка сайту виглядає наступним чином, як представлено на рис. 3.1.

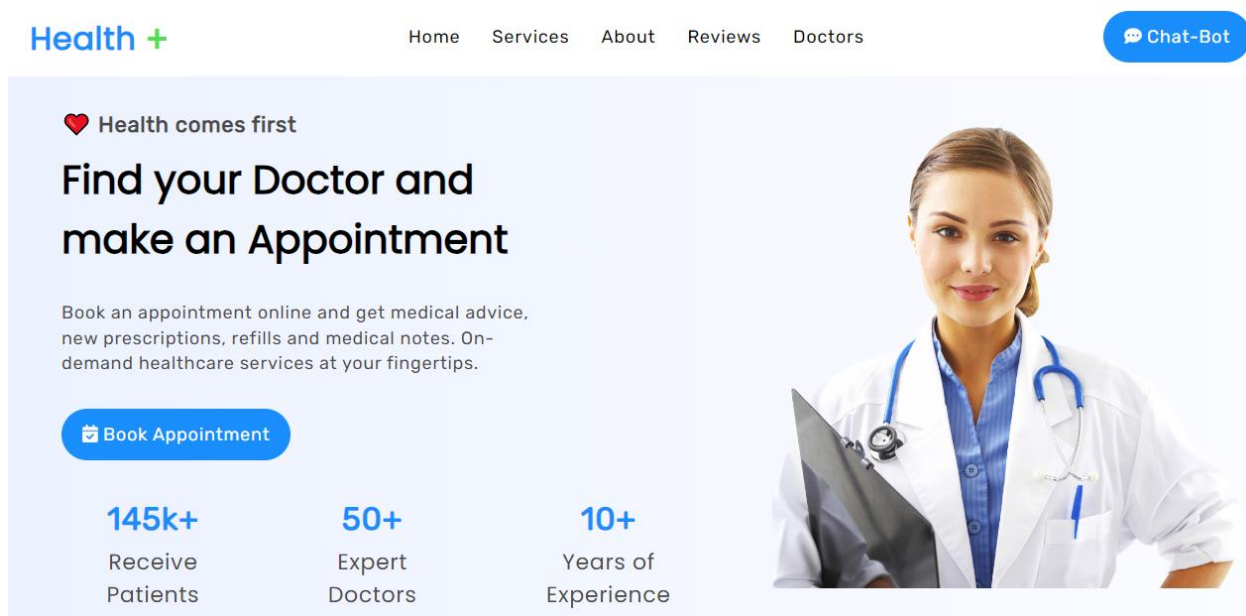


Рис. 3.1. Головна сторінка веб-сайту

У головному меню надано посилання на інші розділи сайту, на які користувач переходить при натисканні на них. Наприклад, при натисканні на посилання About відкривається розділ з інформацією про клініку, який виглядає, як представлено на рис. 3.2.

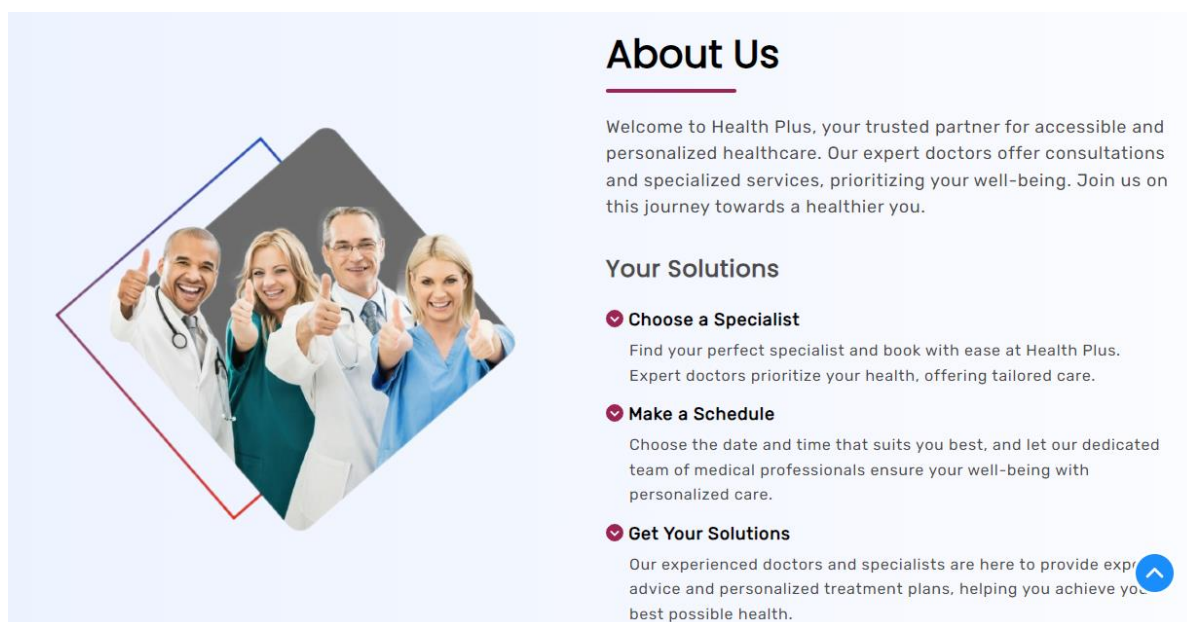


Рис. 3.2. Розділ About

При натисканні на кнопку Book Appointment на головній сторінці відкривається форма для залишення заявки на прийом у лікаря, яка представлена на рис. 3.3.

Health +

Book Appointment Online

* First Name:

* Last Name:

* E-mail:

* Phone Number:

Рис. 3.3. Форма для запису на прийом

Чат-бот, розробка якого представлена у розділі 2, підключається до веб-ресурсу за допомогою бібліотеки Flask на стороні розроблених на мові Python та методу fetch на стороні клієнта, розробленому за допомогою React. Лістинг підключення додатку представлено на рис. 3.4-3.5.

```
@app.route("/chatbot", methods=['POST'])
def get_bot_response():
    data = request.json['message']
    answer = str(data)
    response = str(bot.get_response(answer))
    return jsonify(response)
```

Рис. 3.4. Лістинг підключення (Python)

```
fetch(baseUrl, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({ 'message': request })
})
.then(res : Response => res.json())
.then(res => setResponse(res))
.catch(err => console.log(err))
```

Рис. 3.5. Лістинг підключення (React)

Для користувача чат-бот доступний при натисканні на кнопку Chat-Bot у головному меню. Приклад роботи чат-боту представлено на рис. 3.6.

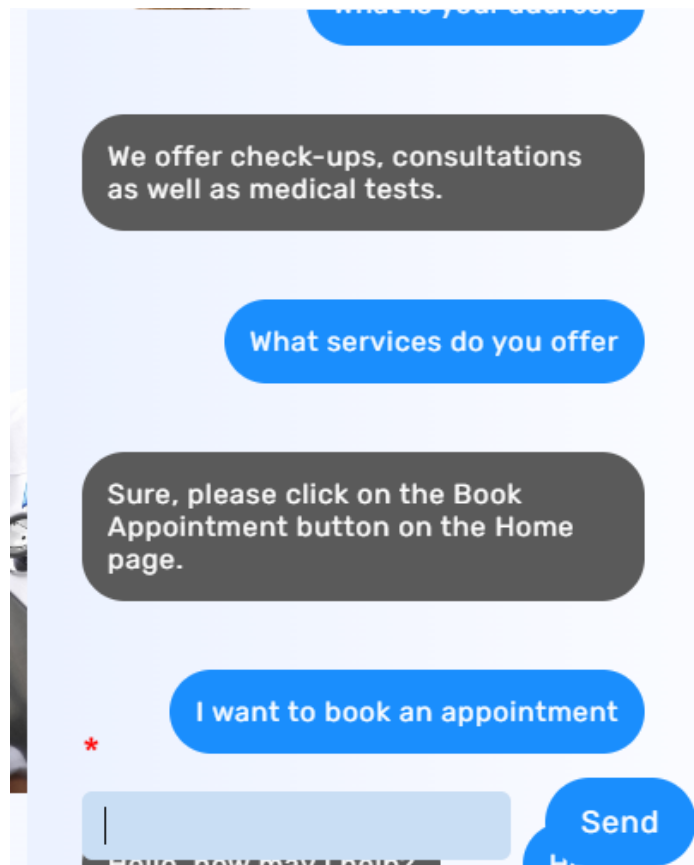


Рис. 3.6. Приклад роботи чат-боту

3.2. Аналіз необхідності використання чат-ботів

Інноваційні рішення, такі як надання клієнтам персоналізованого досвіду за допомогою застосування чат-ботів, забезпечують такі переваги: зручність і ефективність для клієнтів, скорочення часу обслуговування клієнтів, більше інформації, підвищення продуктивності та задоволеності клієнтів.

Чат-боти або віртуальні помічники змінили маркетинговий ландшафт завдяки підвищенню автоматизації, повсюдній персональній допомозі, цілодобовому обслуговуванню клієнтів і гіпертаргетингу. Крім того, як технологія самообслуговування, чат-боти можуть імітувати людську розмову за допомогою введення тексту, аудіовведення або обох, і є формою унікальної технології обробки природної мови, яка забезпечує виняткову точність і продуктивність, що призводить до збільшення використання клієнтами та задоволення.

Усе, що робить організація, сприяє покращенню досвіду клієнтів, а клієнти цінують досвід. Зросла кількість технологічних точок контакту між клієнтами та організаціями. Клієнти можуть отримати доступ до організації через багато взаємопов'язаних інтерфейсів, і будь-яка окрема точка взаємодії, яка не забезпечує очікуваного результату, може мати негативний вплив на досвід і ставлення клієнтів.

Взаємодія з клієнтом є важливою складовою загального продукту чи послуги, що купується. Його важливість також зумовлена роллю, яку він відіграє у задоволенні клієнтів і, у свою чергу, у можливості досягнення конкурентної переваги. Чат-бот дозволяє клієнту надавати послуги без будь-якого контакту з обслуговуючим персоналом, таким чином покладаючи відповідальність на клієнта, створюючи нерозривність між клієнтом і технологією. Ця взаємодія означатиме новий і унікальний досвід клієнта, оскільки «зустріч із обслуговуванням» відрізняється від особистої зустрічі. Багато клієнтів повідомляють, що вони відчувають тривогу, якщо онлайн-досвід не є бездоганим. Таким чином, чи оцінить клієнт досвід як

сприятливий чи несприятливий, залежить від досвіду, який може бути утилітарним (швидкість, ефективність) або неутилітарним (емоційні аспекти). Клієнт оцінює ці враження як позитивні чи негативні, хороші чи погані, як приємні чи неприємні або як задоволення та насолоду. Коли оцінка технологічного досвіду відповідає очікуванням клієнта, результатом є задоволення.

Щоб розглянути поведінку користувачів чат-ботів, дослідники вивчили, як користувачі взаємодіють із чат-ботами як у лабораторних, так і в реальних умовах. Наприклад, Хілл та ін. [26] порівняли взаємодію користувача з чат-ботом із взаємодією користувача під час чатів з іншими користувачами та виявив, що спілкування з чат-ботами передбачало довші розмови з більшою кількістю та коротшими повідомленнями та меншою насиченістю, ніж те, що було виявлено в чатах з іншими користувачами. Порчерон та ін. [35] досліджували реальне використання голосового розмовного агента та виявив, що контекст реального світу створює проблеми для розмови та призводить до частих збоїв у взаємодії чат-бота. Подібні дослідження дають зрозуміти, як користувачі поведуться під час взаємодії з чат-ботами або у відповідь на них. Таким чином, вони надають цінні вказівки щодо дизайну взаємодії чат-ботів, наприклад, щоб допомогти пом'якшити проблеми зручності використання та покращити розмовне відновлення.

Окрім досліджень щодо використання чат-ботів, також зростає інтерес до вивчення сприйняття та уподобань користувачів щодо чат-ботів. Наприклад, Брандтцег і Фьолстад [10] досліджували мотивацію користувачів для використання чат-ботів і виявили, що продуктивність була основним рушієм використання, за яким йдуть розваги, соціальні цілі та інтерес до чат-ботів як до нової технології чи інтерфейсу користувача. Тіес та ін. [39] досліджували переваги користувачів для різних персон чат-ботів і виявили, що персони, які відображають як продуктивність, так і залученість, є кращими для користувачів. Люгер і Селлен [31] досліджували враження користувачів від використання голосових чат-ботів і виявили, що користувачам важко

правильно розуміти можливості чат-ботів і ефективно взаємодіяти з ними; вони припустили, що гра може бути корисною точкою входу для вивчення функцій чат-бота та ефективної взаємодії.

Відповідне уявлення про використання чат-бота та досвід користувача також можна почерпнути з літератури з оцінки діалогових систем, наприклад, з точки зору структур і метрик для оцінки діалогу [30][40], які націлені на такі конструкції, як ефективність, ефективність і задоволеність користувачів завданнями. орієнтовані системи, а також відповідність і схожість на людину для систем, орієнтованих на світську розмову [14]. Крім того, виклики систем діалогу, такі як премія Alexa [28], де команди змагаються за розробку голосових чат-ботів для залучення соціальних взаємодій, суттєво сприяють покращенню нашого розуміння сприйняття та уподобань користувачів щодо чат-ботів. Однак у поточній дослідницькій літературі бракує розуміння того, як прагматичні та гедонічні атрибути, окремо чи в поєднанні, сприяють загальному досвіду користувача чат-бота.

Використання чат-ботів і досвід користувачів вивчалися в широкому діапазоні контекстів, включаючи освіту [19], психічне здоров'я [15] та інформаційні послуги [13]. У таких дослідженнях суб'єктивне сприйняття чат-ботів у користувачів розглядалося як релевантні критерії оцінки. Наприклад, Фітцпатрик та ін. [15] досліджували найкращий і найгірший досвід користувачів під час використання ними терапевтичного чат-бота. Крутцен та ін. [13] запитали користувачів інформаційного чат-бота для молоді про те, чи хочуть вони шукати інформацію в чат-боті чи з інших джерел. Чого зараз не вистачає, так це досліджень взаємодії з чат-ботами, які охоплюють ширший набір контекстів, щоб отримати розуміння аспектів взаємодії з чат-ботами, які є відносно стабільними в різних контекстах. Цікаво, що значна частина досліджень досвіду роботи з чат-ботами проводиться як дослідження Wizard of Oz; тобто учасники взаємодіють не з реальним чат-ботом, а з людиною, яка симулює чат-бота [6][12][39], що говорить про труднощі дослідження досвіду користувача для концепцій чат-ботів на ранніх стадіях розробки. Це, у свою

чергу, підкреслює важливість проведення досліджень взаємодії з чат-ботами, які стали доступними для користувачів.

Теорії досвіду користувача з інтерактивними системами припускають, що прагматичні та гедонічні аспекти є критичними [29]. Здається, це також стосується взаємодії з чат-ботом. Зокрема, важливі прагматичні аспекти досвіду користувача. Хороший досвід роботи з чат-ботом створюється завдяки корисній та ефективній взаємодії. У той же час користувачі також визнають важливість гедонічних аспектів користувацького досвіду. Зокрема, користувачі підкреслюють розважальну цінність чат-ботів, хоча потенційно надихаючу цінність чат-ботів і новизну взаємодії з чат-ботами деякі також згадують як потенційно корисні для роботи з чат-ботами. Потенційна вигода від поєднання сильних прагматичних і гедонічних атрибутів у чат-ботах не є дивною, якщо розглядати її з точки зору структури Хассензала [25] – оскільки конструкції, сильні в обох групах атрибутів, вважаються бажаними.

Ці висновки перегукуються з попередньою роботою щодо мотивації та переваг користувачів щодо використання чат-бота. Брандтцег і Фьолстад [10] у своєму дослідженні мотивації користувачів чат-бота виявили, що головним мотиватором для більшості користувачів є продуктивність, а потім розваги. Тіес та ін. [39] виявили, що потенційні користувачі чат-ботів віддають перевагу особистості чат-бота, яка відображає як продуктивність, так і залученість. Для постачальників послуг чат-ботів може бути корисним розробляти як прагматичні, так і гедонічні атрибути чат-ботів, за умови, що область застосування підходить для цього.

Тому, отримані уроки, що стосуються, зокрема, орієнтованих на завдання чат-ботів [18].

1. У орієнтованих на завдання чат-ботах корисність є головною. Для орієнтованих на завдання додатків чат-ботів вирішення проблем користувачів і допомога користувачам у досягненні їхніх цілей ефективним і дієвим способом є ключовими для забезпечення якісної роботи з чат-ботами. Для стійкого інтересу важливо надавати цінну допомогу та підтримку, а практично

для всіх додатків чат-ботів важливо правильно інтерпретувати наміри користувача та надавати адекватні відповіді. Незважаючи на те, що чат-боти все ще є новою формою інтерактивної системи, постачальникам послуг важливо ретельно стежити за тим, щоб їхні чат-боти служили запланованим цілям і щоб ці цілі оцінювалися їхніми користувачами як корисні.

2. Гедонічні атрибути можуть покращити взаємодію з користувачем у орієнтованих на завдання чат-ботах. Для багатьох програм, орієнтованих на завдання чат-ботів, взаємодія з користувачем може бути посилена шляхом поєднання прагматичних і гедонічних атрибутів чат-ботів. Хоча дуже корисний чат-бот може забезпечувати хорошу взаємодію з користувачем, цей досвід можна ще більше посилити шляхом уважного включення вмісту та характеристик чат-бота, які сприймаються як приємні, спонукальні або грайливі.

3. Звіти користувачів є цінними. Зрозуміти, як користувачі працюють із чат-ботами, складно. Тим не менш, розуміння такого досвіду має вирішальне значення для постачальників послуг чат-ботів, щоб посилити сприйняття чат-ботів серед населення. Завдяки представленому дослідницькому підходу ми продемонстрували доцільність і переваги збору звітів користувачів за допомогою опитувальника на основі техніки критичних інцидентів. Ми сподіваємося, що це стане прикладом того, як постачальники послуг можуть підійти до збору якісних звітів користувачів, які надають настільки необхідну багату інформацію про досвід роботи чат-бота.

4. Різні користувачі мають різні потреби. Взаємодія природної мови, яка виникає в чат-ботах, робить їх дуже придатними для персоналізації. Прикладом цього є наш висновок про те, що прагматичні та гедонічні атрибути чат-бота мають різне значення для користувачів чат-бота різних вікових груп. Однак можливість персоналізації чат-ботів реалізована недостатньо. Навпаки, поточні чат-боти зазвичай виявляють однакову індивідуальність і надають однаковий вміст незалежно від характеристик користувача. Постачальники послуг чат-ботів можуть отримати вигоду від інвестицій у розуміння потреб

своїх різних груп користувачів і налаштування чат-ботів, які можуть відповідним чином адаптуватися.

Очевидно, що чат-боти впливають як на організації, так і на клієнтів; однак, зосередившись на взаємодії клієнтів із чат-ботами, стало зрозуміло, що клієнтам потрібні чат-боти, щоб вони були корисними та простими у використанні. Вони очікують, що чат-боти виконають своє основне завдання та перевершать їхні програми, альтернативні пошукові системи чи веб-сайти, пропонуючи різноманітні та (або) розширені функції. Щоб чат-бот був корисним, він також повинен повідомляти про свої функції та перевіряти придатність домену. Клієнти, які взаємодіють із чат-ботами, хочуть отримувати до них доступ будь-де та коли їм це зручно. Чат-боти пропонують багато можливостей, оскільки активні користувачі Інтернету вважають, що нові технології пропонують більше можливостей, ніж ризиків. Клієнти, що розвивається, цінують досвід, а позитивний досвід приведе до задоволених клієнтів, які сприяють стійкості, зростанню та процвітанню будь-якої організації.

Висновки до розділу 3.

В межах роботи над третім розділом було розгорнуто веб-ресурс та продемонстровано, як було об'єднано розроблений у попередньому розділі чат-бот з розробленим веб-ресурсом, а також продемонстровано роботу сайту та чат-боту. Також було проведено дослідження необхідності використання чат-ботів для веб-ресурсів загалом та їхні перспективи у майбутньому.

ВИСНОВКИ

Під час кваліфікаційної роботи було проведено дослідження технології розробки чат-ботів. Метою дослідження був аналіз методів та засобів для розробки якісного та зручного для користувача інформаційного ресурсу для клініки/лікарні.

Для досягнення даної мети було вирішено наступні задачі:

- 1) проаналізовано наявні веб-ресурси клінік та лікарень, обґрунтовано вибір напрямку та інструментів розробки;
- 2) проаналізовано наявні інструменти для розробки чат-ботів та обґрунтовано вибір інструменту;
- 3) визначено архітектуру та узагальнену структуру веб-сайту, спроектовано та реалізовано структурні складові та алгоритми роботи веб-сайту;
- 4) визначено архітектуру та узагальнену структуру чат-боту, спроектовано та реалізовано структурні складові та алгоритми роботи чат-боту;
- 5) реалізовано інструмент для розробки чат-боту та проведено його тренування;
- 6) проведено аналіз необхідності використання чат-ботів при розробці веб-ресурсів.

В межах дослідження було розглянуто концепції, методи та алгоритми розробки веб-сайтів для клінік, а також для розробки чат-ботів. Було також проаналізовано існуючі приклади веб-ресурсів для лікарень та клінік.

В ході виконання другого розділу було розроблено чат-боту без використання фрейму ChatterBot та наведено повний опис джерела даних, що використовується для навчання чат-боту, і описано його структуру, а також представлено і описано реалізовані методи, алгоритми, використані бібліотеки для розробки та навчання чат-боту. Також було проведено візуальний аналіз, описано його результати та за отриманими результатами зроблено висновки про роботу чат-боту. Після чого було розроблено чат-бот за допомогою

фрейму ChatterBot, який і буде використовуватись для готового веб-ресурсу, описано реалізовані методи та процес розробки і навчання чат-боту, та представлено приклад його роботи.

В межах роботи над третім розділом було розгорнуто веб-ресурс та продемонстровано, як було об'єднано розроблений у попередньому розділі чат-бот з розробленим веб-ресурсом, а також продемонстровано роботу сайту та чат-боту. Також було проведено дослідження необхідності використання чат-ботів для веб-ресурсів загалом та їхні перспективи у майбутньому.

В підсумку було проведено дослідження концепцій та технологій розробки чат-ботів, розроблено власний чат-бот, який був підключений до розробленого веб-ресурсу, а також проведено дослідження необхідності та актуальності використання чат-ботів.

Список використаних джерел

1. Скріпченко Д. Г., Вакалюк Т. А. Розгляд AI чат-ботів. Тези відповідей V Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення" 01-02 грудня 2022 р.
2. Скріпченко Д. Г., Вакалюк Т. А. Розробка інформаційної системи клінік та лікарень. Тези III Всеукраїнської науково-практичної Інтернет-конференції «Сучасні комп'ютерні та інформаційні системи і технології» 12-19 грудня 2022 р.
3. Скріпченко Д. Г., Вакалюк Т.А. Розгляд AI чат-ботів. Тези XIII Міжнародної науково-технічної конференції 30-31 березня 2023 р.
4. Adamopoulou, E. and Moussiades, L. (1970) An overview of chatbot technology, SpringerLink. Springer International Publishing: [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-030-49186-4_31
5. Ahmad N., Hamid M., Zainal A. et al (2020) Review of Chatbots Design Techniques [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/profile/Nahdatul-Akma-Ahmad/publication/327097910_Review_of_Chatbots_Design_Techniques/links/5b77cf3e4585151fd11cd905/Review-of-Chatbots-Design-Techniques.pdf
6. Avula, S., Chadwick, G., Arguello, J., Capra, R. (2018) Searchbots: user engagement with chatbots during collaborative search. In: Proceedings of the 2018 conference on human information interaction & retrieval. ACM, New York, NY.
7. Batish, R. (2018) Voicebot and chatbot design flexible conversational interfaces with Amazon Alexa, Google Home, and Facebook Messenger. Birmingham: Packt.
8. Bhandari, U., Neben, T., Chang, K., Chua, W.Y. (2017) Effects of interface design factors on affective responses and quality evaluations in mobile applications. Comput Hum Behav.
9. Boduch, A. (2020) React and React Native. A. Boduch, R. Derks. p. 526.

10. Brandtzaeg, P.B., Følstad, A. (2017) Why people use chatbots. In: Proceedings of the international conference on internet science. Springer, Cham, Switzerland, pp 377–392.
11. Casciaro, M. (2016) Node.js Design Patterns. M. Casciaro, L. Mammino., p. 777.
12. Chaves, A.P., Gerosa, M.A. (2018) Single or multiple conversational agents?: An interactional coherence comparison. In: Proceedings of the 2018 CHI conference on human factors in computing systems. ACM, New York, NY, paper no. 191.
13. Crutzen, R., Peters, G.J.Y., Portugal, S.D., Fisser, E.M., Grolleman, J.J. (2011) An artificially intelligent chat agent that answers adolescents' questions related to sex, drugs, and alcohol: an exploratory study. *J Adolesc Health*, 48(5). pp. 514–519.
14. Deriu, J., Rodrigo, A., Otegi, A. et al. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54, 755–810 (2021). <https://doi.org/10.1007/s10462-020-09866-x>
15. Fitzpatrick, K.K., Darcy, A., Vierhile, M. (2017) Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (Woebot): A randomized controlled trial. *JMIR Ment Health* 2017;4(2):e19. doi: 10.2196/mental.7785.
16. Freed, A.R. (2021) Conversational AI: Chatbots that work. Shelter Island, NY: Manning.
17. Freeman, A. (2019) Pro React 16 / Adam Freeman. p. 768.
18. Følstad, A., Brandtzaeg, P.B. Users' experiences with chatbots: findings from a questionnaire study. *Qual User Exp* 5, 3 (2020). [Электронный ресурс] - Режим доступа до ресурсу: <https://link.springer.com/article/10.1007/s41233-020-00033-2#Sec26>
19. Fryer, L.K., Ainley, M., Thompson, A., Gibson, A., Sherlock, Z. (2017) Stimulating and sustaining interest in a language course: an experimental

comparison of chatbot and human task partners. *Comput Hum Behav*, 75. pp 461–468.

20. Gallant, L. (2017) User-Centric Hospital Web Sites: A Case for Trust and Personalization [Электронный ресурс] – Режим доступа до ресурсу: <https://www.jstor.org/stable/10.2979/esj.2007.5.2.5>

21. Geron, A. (2020) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition). Aurélien Géron., p. 820.

22. Gibson, A. (2017) Deep Learning: A Practitioner's Approach. A. Gibson, J. Patterson. p. 530.

23. Gruca, T. (2020) Hospital web sites: Promise and progress [Электронный ресурс] – Режим доступа до ресурсу: <https://www.sciencedirect.com/science/article/pii/S0148296302003491>

24. Hasnain, M.(2020) Learning and programming challenges of React.js open-source framework [Электронный ресурс] – Режим доступа до ресурсу: <https://www.emerald.com/insight/content/doi/10.1108/LHTN-05-2023-0088/full/html>

25. Hassenzahl, M. (2018) The thing and I: understanding the relationship between user and product. In: Blythe M, Monk A (eds) *Funology 2. Human–computer interaction series*. Springer, Cham, pp 301–313.

26. Hill, J., Ford, W.R., Farreras, I.G. (2015) Real conversations with artificial intelligence: a comparison between human–human online conversations and human–chatbot conversations. *Comput Hum Behav* 49, pp 245–250.

27. Hospital Websites: From the Information Repository to Interactive Channel [Электронный ресурс] – Режим доступа до ресурсу: <https://www.sciencedirect.com/science/article/pii/S1877050919321945>

28. Khatri, C., Hedayatnia, B., Venkatesh, A., Nunn, J., et al (2018) Advancing the state of the art in open domain dialog systems through the Alexa Prize. 2nd Proceedings of Alexa Prize. Pp. 1.-8. [Электронный ресурс] – Режим доступа до ресурсу: <https://arxiv.org/ftp/arxiv/papers/1812/1812.10757.pdf>

29. Law, E.L.-C., Roto, V., Hassenzahl, M., Vermeeren, A.P., Kort, J. (2009) Understanding, scoping and defining user experience: a survey approach. In: Proceedings of the SIGCHI conference on human factors in computing system. ACM, New York, NY, pp 719–72823.

30. Lowe, R., Noseworthy, M., Serban, I.V., Angelard-Gontier, N., Bengio, Y., Pineau, J. (2017) Towards an automatic Turing test: learning to evaluate dialogue responses. In: Proceedings of the 55th annual meeting of the Association for Computational Linguistics. ACL, Stroudsburg, PA [Электронный ресурс] – Режим доступа до ресурсу: <https://www.aclweb.org/anthology/P17-1103/>

31. Luger, E., Sellen, A. (2016) Like having a really bad PA: the gulf between user expectation and experience of conversational agents. In: Proceedings of the SIGCHI conference on human factors in computing system. ACM, New York, NY, pp 5286–5297.

32. Nguyen, Q. (2018) Understanding User Interactions with a Chatbot: A Self-determination Theory Approach [Электронный ресурс] – Режим доступа до ресурсу:

https://web.archive.org/web/20200322163359id_/https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1419&context=amcis2018.

33. Muldowney, O. (2017) Chatbots An Introduction And Easy Guide To Making Your Own. Dublin, Ireland Curses & Magic. [Электронный ресурс] – Режим доступа до ресурсу: <https://books.google.com.ua/books?id=l8cLvwEACAAJ>.

34. Papa, J. (2021) PyTorch Pocket Reference: Building and Deploying Deep Learning Models. Joi Papa., p. 310.

35. Porcheron, M., Fischer, J.E., Reeves, S., Sharples, S. (2018) Voice interfaces in everyday life. In: Proceedings of the 2018 CHI conference on human factors in computing systems. ACM, New York, NY, paper no. 640.

36. Schmidhuber, J. (2015) "Deep Learning in Neural Networks: An Overview". Neural Networks. 61, pp 85–117.

37. Semaan, P. (2012). Natural Language Generation: An Overview. *Journal of Computer Science & Research (JCSCR)*-ISSN, pp 50-57.
38. Sugondo, A.F. and Bahana, R. (2020) 'IOP Conference Series: Earth and Environmental Science', in IOP Science. [Электронный ресурс] – Режим доступа до ресурсу: <https://iopscience.iop.org/article/10.1088/1755-1315/426/1/012168>.
39. Thies, I.M., Menon, N., Magapu, S., Subramony, M., O'Neill, J. (2017) How do you want your chatbot? An exploratory Wizard-of-Oz study with young, urban Indians. In: *IFIP conference on human–computer interaction*. Springer, Cham, Switzerland, pp 441–459.
40. Walker, M.A., Litman, D.J., Kamm, C.A., Abella, A. (1997) PARADISE: a framework for evaluating spoken dialogue agents. In: *Proceedings of 35th annual meeting of the Association for Computational Linguistics and 8th conference of the European chapter of the Association for Computational Linguistics*. ACL, Stroudsburg, PA, pp 271–280.
41. Yampolskiy, R. V. (2013) Turing Test as a Defining Feature of AI-Completeness. In *Artificial Intelligence, Evolutionary Computation and Metaheuristics (AIECM)* pp. 3-17.

ДОДАТКИ

```
import nltk

from nltk.stem.lancaster import LancasterStemmer

import numpy as np

import tflearn

import tensorflow as tf

import datetime

import json

import pickle

import random


nltk.download("punkt")

with open('intents.json') as intents:
    data = json.load(intents)


stemmer = LancasterStemmer()


words = []
labels = []
x_docs = []
y_docs = []


for intent in data['intents']:
    for pattern in intent['patterns']:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        x_docs.append(wrds)
        y_docs.append(intent['tag'])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])
```

```

words = [stemmer.stem(w.lower()) for w in words if w not in "?"]
words = sorted(list(set(words)))
labels = sorted(labels)

training = []
output = []
out_empty = [0 for _ in range(len(labels))]

for x, doc in enumerate(x_docs):
    bag = []
    wrds = [stemmer.stem(w) for w in doc]
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)

    output_row = out_empty[:]
    output_row[labels.index(y_docs[x])] = 1

    training.append(bag)
    output.append(output_row)

training = np.array(training)
output = np.array(output)

net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, len(output[0]), activation='softmax')
net = tflearn.regression(net)

```

```

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

model = tflearn.DNN(net, tensorboard_verbose=3, tensorboard_dir=log_dir)
model.fit(training, output, n_epoch=500, batch_size=8, show_metric=True)

def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]
    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for s_word in s_words:
        for i, w in enumerate(words):
            if w == s_word:
                bag[i] = 1
    return np.array(bag)

def chat():

    while True:
        inp = input("\n\nYou: ")
        if inp.lower() == 'quit':
            break

        results = model.predict([bag_of_words(inp, words)])

        results_index = np.argmax(results)

        tag = labels[results_index]

        for tg in data['intents']:

            if tg['tag'] == tag:

```

```
responses = tg['responses']  
print("Bot:\t" + random.choice(responses))  
  
chat()
```

Додаток Б

Файл medchatbot.py

```
from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer, ChatterBotCorpusTrainer

bot = ChatBot('MedBot', read_only=True,
               preprocessors=[ 'chatterbot.preprocessors.convert_to_ascii',
                               'chatterbot.preprocessors.unescape_html',
                               'chatterbot.preprocessors.clean_whitespace'],
               logic_adapters=[
                   {
                       'import_path': 'chatterbot.logic.BestMatch',
                       'default_response': 'Sorry, I am unable to process your
request.',
                       'maximum_similarity_threshold': 0.90
                   }
               ],)

trainer = ListTrainer(bot)
data = open('training_data.txt').read()
conversations = data.strip().split('\n')

trainer.train(conversations)

def chatbot():
    print("Welcome to the ChatBot Service for Med! How can I help you? (Enter
'Bye' to exit)")
    while True:
        request = input('You:')
        if request == 'Bye' or request == 'bye':
            print('ChatBot: It was great talking to you!')
            break
        else:
            response = bot.get_response(request)
            print('ChatBot:', response)
```

Файл server.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from medchatbot import bot

app = Flask(__name__)
CORS(app)

@app.route("/chatbot", methods=['POST'])
def get_bot_response():
    data = request.json['message']
    answer = str(data)
    response = str(bot.get_response(answer))
    return jsonify(response)

if __name__ == "__main__":
    app.run()
```