Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної роботи
другий(магістерський)
(рівень вищої освіти)

на тему: **РОЗРОБКА ПЕРСОНАЛЬНОГО ПОМІЧНИКА ЯК ЕЛЕМЕНТА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ**

Виконав: студент 6 курсу, групи 6дТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Кузьмін А.В.
(прізвище та ініціали)
Керівник д.ф.-м.н, проф. Миронцов М.Л.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Полтава – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ « ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
спеціальність 122«Комп'ютерні науки»на тему
«РОЗРОБКА ПЕРСОНАЛЬНОГО ПОМІЧНИКА ЯК ЕЛЕМЕНТА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ»

Студента групи 6дТН Кузьміна Артура Валентиновича

Керівник роботи
д.ф.-м.н, професор
Миронцов М.Л.

Завідувач кафедри
кандидат фізико-математичних наук, Двірна О.А.

Полтава – 2025

РЕФЕРАТ

Кваліфікаційна робота магістра: 67 сторінок, 15 рисунків, 24 джерела.

Об’єкт дослідження: процес розробки системи рекомендацій для автоматизації надання порад.

Предмет дослідження: методи машинного навчання для створення системи рекомендацій на основі симптомів та відповідних рекомендацій.

Мета роботи: проектування та розробка системи рекомендацій, яка дозволяє автоматично надавати корисні поради користувачам на основі введених ними симптомів.

Методи: Для реалізації проекту було використано кілька методів: проведено аналіз літератури для вивчення сучасних підходів до розробки систем рекомендацій та технологій обробки текстових даних; за допомогою методу порівняння обрано оптимальні алгоритми машинного навчання, зокрема Random Forest для класифікації; метод моделювання дозволив побудувати модель на основі навчальних даних після їх попередньої обробки; метод експерименту використовувався для тестування моделі на валідаційних даних з метою перевірки точності та якості рекомендацій; метод аналізу даних застосовано для обробки результатів тестування, оптимізації моделі та вдосконалення системи рекомендацій.

Ключові слова: система рекомендацій, машинне навчання, класифікація, Random Forest, автоматизація, текстові дані, енкодери, моделювання, Python, телеграм-бот.

ABSTRACT

Bachelor's qualification work: 67 p., 15 pictures, 24 sources.

Object of research: The process of developing a recommendation system for automating the provision of advice.

Subject of the research: Machine learning methods for creating a recommendation system based on symptoms and corresponding recommendations.

Purpose: To design and develop a recommendation system that automatically provides useful advice to users based on their input symptoms.

Methods: a literature review was conducted to explore modern approaches to the development of recommendation systems and text data processing technologies; the comparison method was used to select optimal machine learning algorithms, specifically Random Forest for classification; the modeling method facilitated the creation of a model based on preprocessed training data; the experimental method was applied to test the model on validation data to verify its accuracy and the quality of recommendations; the data analysis method was used to process the testing results, optimize the model, and improve the recommendation system.

Keywords: recommendation system, machine learning, classification, Random Forest, automation, text data, encoders, modeling, Python, Telegram bot.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким зростанням обсягів інформації, що вимагає від людей швидкого та обґрунтованого прийняття рішень у різних сферах діяльності. Системи підтримки прийняття рішень (СППР) стають важливими інструментами, що дозволяють мінімізувати помилки, знижувати витрати часу і підвищувати ефективність роботи. У цьому контексті персональні помічники відіграють значну роль, забезпечуючи користувачів релевантними даними, автоматизуючи рутинні завдання та сприяючи швидкому прийняттю рішень. Поєднання СППР із сучасними технологіями взаємодії, такими як чат-боти, відкриває нові горизонти для оптимізації процесів у бізнесі, освіті, управлінні та інших галузях.

Telegram, як одна з найпопулярніших платформ для обміну повідомленнями, надає ефективний інструментарій для створення інтерактивних чат-ботів. Завдяки відкритому API, простоті інтеграції та широким функціональним можливостям, Telegram-боти використовуються для автоматизації різноманітних завдань: від бронювання послуг до обробки даних у реальному часі. Їх гнучкість і зручність у використанні роблять їх ідеальними інтерфейсами для реалізації персональних помічників у рамках СППР.

Актуальність даного дослідження зумовлена потребою у створенні інструментів, які б об'єднували функціональність сучасних систем підтримки прийняття рішень із доступністю та зручністю Telegram-ботів. Розробка персонального помічника у формі Telegram-бота дозволить не лише автоматизувати рутинні завдання, а й створити основу для впровадження більш складних інтелектуальних алгоритмів, що підвищать якість прийняття рішень і зроблять ці системи доступними для широкого кола користувачів.

Метою дослідження є розробка персонального помічника у вигляді Telegram-бота як елемента системи підтримки прийняття рішень для автоматизації процесів обробки даних і взаємодії з користувачами.

Для досягнення поставленої мети необхідно виконати такі завдання дослідження:

− провести аналіз існуючих систем підтримки прийняття рішень і визначити їх основні функціональні вимоги;

− дослідити можливості Telegram API для створення інтерактивного інтерфейсу персонального помічника;

− спроєктувати архітектуру системи персонального помічника з урахуванням її інтеграції зі сторонніми джерелами даних;

− розробити алгоритми, що реалізують функціональність прийняття рішень та обробки інформації;

− реалізувати серверну частину Telegram-бота з використанням сучасних фреймворків та мов програмування;

− провести тестування функціональних можливостей Telegram-бота та оцінити його ефективність у вирішенні поставлених завдань;

− розробити технічну документацію, що описує принципи роботи бота та особливості його використання;

− проаналізувати перспективи вдосконалення системи та можливості її інтеграції з іншими технологіями.

Об’єкт дослідження – це процес ппроцес автоматизації прийняття рішень за допомогою інформаційних систем.

Предмет дослідження – методи та засоби розробки персонального помічника у вигляді Telegram-бота для підтримки прийняття рішень.

У роботі використано теоретичні методи для аналізу систем підтримки прийняття рішень та огляду можливостей Telegram API. Для розробки персонального помічника застосовано методи системного аналізу та проєктування програмного забезпечення, включаючи модульне проєктування архітектури системи. Реалізація алгоритмів прийняття рішень базувалася на використанні алгоритмічного підходу та мов програмування Python. Тестування функціональності Telegram-бота здійснювалося за допомогою методів юніт-

тестування та інтеграційного тестування, а оцінка ефективності — з використанням методів порівняльного аналізу.

Практична значимість роботи полягає у створенні персонального помічника у вигляді Telegram-бота, який може бути використаний для автоматизації прийняття рішень у різних сферах. Розроблений бот забезпечує зручний інтерфейс для користувачів, дозволяючи оперативно отримувати релевантну інформацію, аналізувати дані та виконувати рутинні завдання. Крім того, результати роботи можуть бути адаптовані для впровадження в інших інформаційних системах, що сприяє підвищенню їх функціональності та доступності. Отримані розробки також можуть бути використані як основа для подальших досліджень у сфері інтеграції СППР з сучасними комунікаційними платформами.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ТА ЗАСОБІВ АВТОМАТИЗАЦІЇ

1.1 Класифікація та структура сучасних систем підтримки прийняття рішень

Системи підтримки прийняття рішень (СППР) — це інтерактивні інформаційні системи, що допомагають користувачам аналізувати дані, моделювати ситуації та приймати обґрунтовані рішення. Вони використовуються в ситуаціях, де рішення залежить від багатьох факторів, потребує аналізу великої кількості інформації або залучення різних сценаріїв. Основними компонентами СППР є: база даних для зберігання інформації, модельна база для аналізу даних та інтерфейс користувача для взаємодії із системою.

Класифікація СППР проводиться за рівнем автоматизації, типом підтримуваних завдань, способом реалізації, характером обробки інформації та галузевою спрямованістю. Розглянемо різні види цих систем детальніше [1].

Сучасних систем підтримки прийняття рішень за рівнем автоматизації поділяються на пасивні, активні та інтерактивні.

Пасивні СППР забезпечують користувача необхідною інформацією для прийняття рішень, але не надають жодних рекомендацій чи оцінки варіантів. Їх головна мета — збирання, обробка та візуалізація даних, щоб користувач міг самостійно аналізувати ситуацію. Прикладами є системи звітності для моніторингу фінансових або виробничих показників, інформаційні панелі (dashboards), що відображають ключові показники ефективності (KPI), інструменти бізнес-аналітики, які генерують графіки, діаграми або звіти на основі заданих критеріїв [2-4].

Пасивні СППР вимагають від користувача значного досвіду й часу для інтерпретації отриманих даних.

Активні СППР не лише надають інформацію, але й активно пропонують варіанти рішень, базуючись на заданих алгоритмах і критеріях. Вони використовують моделі аналізу, оптимізації та прогнозування для підготовки рекомендацій або оптимальних сценаріїв. Прикладами є системи для управління запасами, які пропонують оптимальний обсяг замовлення, системи для фінансового планування, які рекомендують інвестиційні стратегії залежно від ризиків і цілей, програми для вибору постачальників на основі аналізу вартості, якості та часу доставки [2].

Активні СППР значно скорочують час на аналіз, пропонуючи конкретні варіанти дій. Проте рекомендації можуть бути неточними за умов недостатньо якісних даних або неврахованих параметрів.

Інтерактивні СППР надають користувачу можливість взаємодіяти із системою в реальному часі для уточнення критеріїв, внесення нових даних чи перегляду альтернативних сценаріїв. Вони поєднують в собі функції пасивних та активних СППР, дозволяючи користувачу краще адаптувати процес прийняття рішень до конкретної ситуації. Прикладами таких систем є інтерактивні фінансові планувальники, де користувач може змінювати параметри (наприклад, бюджет чи період інвестування) й одразу отримувати оновлені рекомендації, системи для логістичного планування, які дозволяють користувачу змінювати маршрути, переглядати витрати та прогнозувати терміни доставки, платформи з підтримкою чат-ботів, які ведуть діалог з користувачем, уточнюють запити та надають відповіді або сценарії дій.

Перевагами інтерактивних СППР є те, що такі системи забезпечують гнучкість і персоналізацію рішень, дозволяючи враховувати більше факторів у процесі прийняття рішення. Однак, недоліком є висока складність реалізації та потреба в потужних обчислювальних ресурсах [2].

Кожен рівень автоматизації СППР орієнтований на вирішення специфічних завдань. Пасивні СППР підходять для моніторингу та аналізу даних, активні —

для генерації рекомендацій, а інтерактивні забезпечують повний цикл підтримки прийняття рішень із високим рівнем гнучкості та точності.

За типом підтримуваних завдань СППР поділяються на : СППР для стратегічного планування, СППР для тактичного планування та СППР для оперативного управління.

СППР для стратегічного планування підтримують прийняття рішень, що впливають на довгострокову діяльність організації або підприємства. Вони орієнтовані на високу абстракцію та аналіз глобальних тенденцій, що дозволяє розробляти стратегії розвитку, управління ресурсами та інвестиційними проектами на кілька років вперед. СППР для стратегічного планування використовують моделювання та прогнозування для оцінки можливих сценаріїв розвитку, що дозволяє визначити найбільш ефективні шляхи досягнення довгострокових цілей. Прикладами застосування є розробка стратегії розвитку підприємства на 5–10 років, вибір напрямків інвестиційної діяльності, планування масштабних проектів або розширення на нові ринки.

Особливостями використання цих систем є висока складність моделювання, багато факторів невизначеності, великий обсяг даних для аналізу.

СППР для тактичного планування спрямовані на прийняття рішень середнього терміну, що стосуються оптимізації ресурсів та досягнення цілей у межах кількох місяців чи років. СППР для тактичного планування займаються оптимізацією використання ресурсів, таких як фінанси, робоча сила та матеріали, а також проводять аналіз ризиків і можливих перешкод для досягнення цілей. Вони дозволяють планувати оперативні дії для реалізації стратегічних цілей, наприклад, визначати ефективність кампаній, поліпшувати продуктивність або управляти запасами. Серед прикладів застосування - оптимізація розподілу фінансових ресурсів, планування виробничих потужностей, управління ланцюгами постачання, аналіз ризиків для існуючих проектів. Особливостями є менше невизначеності, ніж у стратегічних СППР, але все ще вимагають комплексного аналізу даних для оцінки варіантів [5].

СППР для оперативного управління підтримують прийняття рішень у реальному часі, дозволяючи швидко реагувати на зміни в оперативній ситуації та забезпечувати безперервність функціонування організації. Вони зазвичай використовуються для вирішення тактичних завдань, що вимагають негайного реагування, таких як управління поточними ресурсами, організація доставки, планування роботи в реальному часі та моніторинг виконання завдань. СППР для оперативного управління допомагають приймати рішення, ґрунтуючись на актуальних даних, і забезпечують ефективність операцій у складних і швидко змінних умовах. Прикладами застосування є управління виробничими процесами в реальному часі, моніторинг стану запасів, управління персоналом в умовах змінної попиту, оптимізація маршрутів доставки товарів. Серед особливостей можна зазначити негайне прийняття рішень, обробка великої кількості даних у реальному часі, висока потреба в точності та швидкості дій [5].

Очевидно, що кожен тип СППР орієнтований на різні рівні управлінських завдань: стратегічне планування фокусується на довгострокових цілях, тактичне планування — на оптимізації ресурсів і ризиків, а оперативне управління дозволяє швидко реагувати на зміни в реальному часі. Вибір типу СППР залежить від специфіки завдань і часу на їх виконання.

За способом реалізації СППР класифікують так : локальні та хмарні системи.

Локальні СППР встановлюються на окремих пристроях (персональних комп’ютерах, серверах) або в межах локальної мережі організації. Вони не потребують постійного підключення до Інтернету, всі дані зберігаються на місцевих серверах або пристроях, що дозволяє обробляти чутливу інформацію в межах організації без зовнішнього доступу. Локальні системи можуть бути менш гнучкими в плані масштабування та доступу до даних із віддалених локацій, але вони забезпечують контроль над безпекою та приватністю інформації. Застосовуються як внутрішні бізнес-аналітичні системи, програми для планування та управління персоналом, обробка фінансових даних. Особливостями є можливість детального контролю за даними, висока безпека, обмеження в масштабуванні та доступності [7].

Хмарні СППР працюють через Інтернет, зберігаючи дані на віддалених серверах або в хмарі. Це дає змогу організаціям обробляти та аналізувати великі обсяги інформації, не потребуючи дорогих апаратних засобів для зберігання та обчислень. Хмарні платформи забезпечують гнучкість в управлінні ресурсами, а також можливість доступу до системи з будь-якої точки світу. Вони добре підходять для масштабних аналізів даних та підтримки великих обсягів інформації, а також для колаборації між різними користувачами. Застосуються як хмарні платформи для обробки великих даних, аналітичні інструменти для прогнозування та управлінських рішень, глобальні бізнес-платформи, такі як Microsoft Power BI або Google Analytics. Серед особливостей модна виокремити легке масштабування, доступність у будь-який час і з будь-якого місця, високі вимоги до інтернет-з'єднання та безпеки [7].

За характером обробки інформації виокремлюють такі СППР : на основі моделювання, на основі аналізу даних, на базі штучного інтелекту.

СППР на основі моделювання використовують моделі для аналізу сценаріїв та прогнозування. Вони допомагають передбачити майбутні результати на основі наявних даних, створюючи різні сценарії для оцінки можливих варіантів розвитку подій. Моделювання може бути математичним, статистичним або економічним, і зазвичай використовується для стратегічного або тактичного планування, коли потрібно вивчити наслідки різних рішень. Застосують для моделювання економічних процесів, прогнозування попиту на продукцію, оцінка впливу змін у стратегії на довгострокові результати. Притаманним цьому виду СППР є створення моделей для різних варіантів розвитку ситуації, високий рівень абстракції та математичних обчислень, використання даних для створення сценаріїв.

СППР на основі аналізу даних зосереджуються на обробці великих обсягів статистичних даних для виявлення закономірностей, трендів та аномалій. Вони використовують алгоритми аналізу даних для надання корисної інформації, яка допомагає приймати обґрунтовані рішення. Такі системи можуть допомогти виявити приховані взаємозв’язки між різними змінними, спрогнозувати

результати та визначити оптимальні стратегії на основі наявних даних. Прикладами застосування є аналіз продажів, вивчення споживчих тенденцій, фінансовий аналіз для визначення інвестиційних можливостей. Особливостями цих систем є використання статистичних методів, виявлення кореляцій, застосування аналітичних інструментів для глибокого аналізу даних.

СППР на базі штучного інтелекту використовують методи штучного інтелекту (ШІ), зокрема нейронні мережі, машинне навчання та глибинне навчання, для прийняття рішень або вдосконалення процесу аналізу. Вони можуть адаптуватися до нових даних і безпосередньо оптимізувати процес прийняття рішень на основі великих обсягів інформації, що надходить у реальному часі. Такий підхід особливо ефективний для складних, багатофакторних завдань, де традиційні методи моделювання чи аналізу даних можуть бути недостатньо точними або ефективними. Такі системи використовують як рекомендаційні системи, аналіз великої кількості даних для прогнозування поведінки користувачів, вдосконалення моделей для управлінських рішень в реальному часі. Особливостями є адаптивність до нових даних, здатність вчитися на основі попередніх результатів, застосування передових технологій ШІ для обробки складних і великих наборів даних.

Кожна класифікація СППР за способом реалізації та характером обробки інформації дає змогу краще визначити їх застосування в залежності від вимог до системи, обсягів даних, необхідної швидкості обробки і рівня складності задач. Локальні системи підходять для обмежених обсягів даних, тоді як хмарні рішення дозволяють працювати з великими масивами інформації, а методи аналізу та ШІ сприяють підвищенню точності й ефективності прийняття рішень.

Серед СППР за галузевою спрямованістю можна виділити: фінансові, медичні, логістичні та ін [5, 9].

Фінансові СППР використовуються для аналізу фінансових даних, управління портфелями інвестицій, оцінки ризиків та прогнозування економічних результатів. Вони допомагають приймати обґрунтовані фінансові рішення, знижуючи ймовірність збитків і максимізуючи прибуток. Ці системи

застосовуються для аналізу даних про ринки, визначення стратегій інвестування, управління активами та пасивами, а також для розробки моделей ризику та рентабельності. Вони також використовуються для автоматизації фінансового обліку та складання звітності.

Прикладами застосування є оцінка інвестиційних ризиків, аналіз вартості активів, побудова прогнозних моделей для фінансових ринків. Особливостями цих систем є обробка великих обсягів фінансових даних, використання моделей для прогнозування поведінки ринку, застосування методів для оптимізації портфелів інвестицій.

Медичні СППР спрямовані на підтримку прийняття рішень у сфері охорони здоров'я, зокрема на етапах діагностики, планування лікування та моніторингу стану пацієнтів. Вони дозволяють лікарям обробляти великий обсяг медичних даних, таких як історії хвороб, аналізи та обстеження, для більш точного визначення діагнозу, призначення лікування та оцінки ефективності терапії. Крім того, медичні СППР використовуються для автоматизації адміністрування лікарняних процесів, управління ресурсами та планування медичних послуг. Галузями застосуванння є діагностика захворювань за допомогою аналізу медичних зображень, розробка індивідуальних планів лікування для пацієнтів, моніторинг ефективності терапії. Особливостями є використання медичних протоколів, адаптація до індивідуальних характеристик пацієнтів, інтеграція з іншими медичними системами (наприклад, електронні медичні картки).

Логістичні СППР спрямовані на оптимізацію логістичних процесів, таких як управління ланцюгами постачань, оптимізація маршрутів доставки, управління запасами та ефективне розподілення ресурсів. Вони дозволяють автоматизувати та покращити процеси управління доставкою товарів, планувати і контролювати транспортні потоки, забезпечувати своєчасність і точність поставок. Ці системи допомагають знизити витрати на транспортування, зменшити час на доставку та забезпечити високу ефективність роботи логістичних компаній. Застосовуються як оптимізація маршруту доставки товарів, управління складів, планування і контроль запасів, аналіз витрат на транспортування. Характерим є використання

алгоритмів для оптимізації маршрутів, інтеграція з системами моніторингу транспорту, аналіз в реальному часі даних про місцезнаходження товарів.

Класифікація СППР за галузевою спрямованістю дозволяє створювати спеціалізовані системи для різних сфер діяльності. Фінансові СППР орієнтовані на управління ризиками та оптимізацію інвестицій, медичні — на точність діагностики та планування лікування, а логістичні — на ефективність управління ланцюгами поставок і транспортування товарів. Кожна з цих систем використовує специфічні методи обробки та аналізу даних, що відповідають вимогам кожної галузі.

Структура сучасних систем підтримки прийняття рішень (СППР) складається з кількох основних компонентів, які взаємодіють між собою для забезпечення ефективного процесу прийняття рішень. Ці компоненти можна поділити на апаратну та програмну частину, а також на спеціалізовані інтерфейси і бази даних. Ось детальний опис основних складових СППР [10]:

1. База даних. База даних є основною складовою СППР, яка зберігає всі необхідні дані для аналізу та прийняття рішень. Вона може містити історичні дані, поточну інформацію, а також зовнішні дані, які використовуються для моделювання або прогнозування. Бази даних можуть бути реляційними або нереляційними, залежно від вимог системи до типу та структури даних. Наприклад, дані про фінансові результати, історія покупок, медичні записи, логістичні маршрути.

2. Моделі прийняття рішень. Моделі прийняття рішень визначають, як будуть оброблятися вхідні дані для отримання оптимальних рішень. Ці моделі можуть бути математичними, статистичними або базуватися на штучному інтелекті. Моделі допомагають системі прогнозувати наслідки різних рішень, аналізувати ризики, оптимізувати ресурси або знаходити найкращі сценарії. Наприклад, моделі для прогнозування попиту, оцінка ризиків, оптимізація виробничих процесів.

3. Інтерфейс користувача. Інтерфейс користувача є важливою складовою системи, оскільки він дозволяє користувачам взаємодіяти з СППР, вводити

вхідні дані, переглядати результати аналізу і отримувати рекомендації. Це може бути графічний інтерфейс, діалогова система або чат-бот, залежно від типу системи. Інтерфейс користувача забезпечує зрозумілий і зручний спосіб взаємодії з системою, що є критично важливим для користувачів, які не мають глибоких технічних знань. Наприклад, веб-інтерфейси, мобільні додатки, голосові системи.

4. Аналізатор даних. Цей компонент відповідає за обробку великих обсягів даних з різних джерел. Він здійснює первинну обробку, фільтрацію, класифікацію та інтеграцію даних, щоб вони могли бути використані в моделюванні та прийнятті рішень. Включає в себе різні методи статистичного аналізу, аналізу даних і машинного навчання. Наприклад, програмне забезпечення для обробки великих даних, системи для аналізу трендів на фінансових ринках, аналіз даних клієнтів у маркетингу.

5. Інструменти для моделювання та прогнозування. Ці інструменти дозволяють створювати математичні або статистичні моделі для прогнозування результатів різних рішень. Вони можуть включати методи математичного програмування, теорії ігор, штучного інтелекту, нейронні мережі, алгоритми оптимізації і т. ін. Наприклад, використання моделей для прогнозування продажів, моделювання фінансових стратегій, оцінка можливих сценаріїв розвитку підприємства.

6. Модуль підтримки рішень. Модуль підтримки рішень відповідає за створення варіантів рішень або рекомендацій, заснованих на даних і моделях, що були введені в систему. Цей компонент може включати автоматичні алгоритми для оцінки і порівняння варіантів, а також для надання рекомендацій користувачеві. Наприклад, рекомендаційні системи для інвестицій, автоматична оптимізація маршрутів у логістиці, алгоритми вибору постачальників.

7. Комунікаційні та інтеграційні модулі. Ці модулі відповідають за інтеграцію СППР з іншими корпоративними системами, такими як CRM, ERP, системи управління запасами, фінансові системи тощо. Вони забезпечують взаємодію з іншими програмними засобами та обмін даними, що дозволяє використовувати

інформацію з різних джерел для прийняття рішень. Наприклад, інтеграція з корпоративними базами даних, обмін даними між різними підсистемами організації, комунікація з іншими аналітичними інструментами.

8. Система моніторингу та оцінки ефективності. Цей компонент надає можливість відслідковувати ефективність рішень, що були прийняті за допомогою СППР. Важливою частиною є зворотний зв'язок, який дозволяє коригувати моделі та стратегії в реальному часі на основі результатів. Наприклад, оцінка результатів після впровадження рекомендацій, моніторинг результативності застосованих стратегій.

Отже, сучасні системи підтримки прийняття рішень мають складну багаторівневу структуру, де кожен компонент виконує свою роль у забезпеченні ефективності процесу прийняття рішень. Взаємодія цих компонентів дозволяє створити потужну і адаптивну систему, здатну працювати з великими обсягами даних, створювати точні прогнози та надавати рекомендації, що підвищує якість прийнятих рішень у різних сферах діяльності.

1.2 Огляд існуючих Telegram-ботів, їх можливостей та обмежень

Telegram-боти стали одним з найпопулярніших інструментів для автоматизації різноманітних процесів у різних сферах діяльності завдяки простоті використання, потужному API та широкому функціоналу. Вони можуть виконувати різноманітні завдання — від надання інформації до інтерактивного спілкування з користувачами, обробки запитів і навіть управління складними системами. Однак, як і будь-які технології, Telegram-боти мають свої можливості та обмеження.

Розглянемо можливості сучасних Telegram-ботів .

Перш за все, це автоматизація процесів. Telegram-боти можуть автоматизувати безліч процесів, таких як надання даних, організація розкладів, нагадування про важливі події, обробка запитів клієнтів і багато іншого. Вони можуть бути інтегровані з різними системами, що дозволяє їм працювати з

великими обсягами даних, виконувати запити до зовнішніх баз даних та надавати корисну інформацію в реальному часі. Наприклад, боти для замовлення товарів, отримання інформації про фінансові операції, оповіщення про новини.

Важливим аспектом є інтерактивне спілкування з користувачем. Завдяки використанню природного мовлення та можливості зберігати контекст діалогу, Telegram-боти можуть вести змістовні бесіди, відповідаючи на запитання користувачів. Вони здатні виконувати роль консультантів, допомагаючи користувачам швидко отримувати відповіді на запитання або здійснювати певні операції. Наприклад, Боти-консультанти для підтримки клієнтів, боти для надання юридичних або медичних консультацій.

Потужним інструментом є інтеграція з іншими сервісами. Боти можуть бути інтегровані з іншими платформами через API, що дозволяє обробляти платежі, реєстраційні форми, онлайнові консультації та інші інтерактивні сервіси. Вони також можуть отримувати та передавати дані в реальному часі з інших програмних засобів. Наприклад, оплата через бот за допомогою платіжних систем, підключення до CRM-системи для управління запитами клієнтів.

Для сучасних іформаційних систем надважливою є можливість масштабованості. Telegram-боти можуть обслуговувати тисячі користувачів одночасно без значних проблем з продуктивністю. Це дає змогу ефективно працювати з великими обсягами однотипних запитів, що особливо важливо для бізнесу та сервісів з високими навантаженнями. Наприклад, Масове оповіщення користувачів про оновлення або нові функції.

Можливість забезпечення безпеки та конфіденційності також є важливо. Telegram надає високий рівень безпеки завдяки шифруванню даних, що робить ботів безпечними для обробки особистої інформації та фінансових транзакцій. Проте, це залежить від того, як сам бот та його розробники реалізують ці функції.

Хоча Telegram-боти мають ряд привабливих можливостей, проте існує низка обмежень, які ми розглянемо далі.

Обмеження по функціоналу. Telegram-боти обмежені певними функціями, які надаються через API. Вони можуть не бути такими гнучкими або

функціональними, як спеціалізовані програми або веб-сайти, особливо коли мова йде про складні або ресурсомісткі операції. Наприклад, обмежена підтримка багатозадачності, складні графічні інтерфейси можуть бути не такими зручними, як у веб-додатках.

Другим обмеженням є залежність від Telegram. Оскільки Telegram-боти працюють на базі платформ Telegram, вони обмежені можливостями цієї платформи. Наприклад, боти не можуть здійснювати певні операції, які вимагають доступу до файлової системи або використовувати функції, недоступні в Telegram (наприклад, зберігання великого обсягу даних). Наприклад, неможливість реалізації складних візуальних інтерфейсів або зберігання великих файлів безпосередньо в чаті.

Слід зазначити також обмеження на інтеграцію з іншими платформами. Хоча Telegram підтримує інтеграцію з іншими сервісами через API, все ж таки є обмеження щодо того, які системи можна інтегрувати з ботами. Для розробки повноцінних рішень може знадобитися значні додаткові ресурси для налаштування інтеграції. Наприклад, проблеми при інтеграції зі старими або специфічними корпоративними системами, які не підтримують API.

Проблеми з обробкою великих обсягів даних. Telegram-боти зазвичай не призначені для обробки великих обсягів даних або складних обчислень в реальному часі. Це може стати проблемою, якщо бот використовується для складних аналітичних або наукових завдань. Наприклад, не можна використовувати бота для глибоких аналітичних операцій або обробки великих наборів даних без додаткових ресурсів.

Обмежена можливість для персоналізації. Через обмеження Telegram-боти можуть бути менш персоналізованими у порівнянні з іншими програмами. Хоча боти можуть вести діалоги та надавати індивідуальні рекомендації, їх можливості в області глибокої персоналізації досить обмежені, порівняно з розширеними системами підтримки клієнтів або ШІ-рішеннями. Наприклад, Можливість для складних персоналізованих рекомендацій обмежена простими алгоритмами.

Отже, Telegram-боти є потужними інструментами для автоматизації процесів, інтерактивного спілкування з користувачами та інтеграції з іншими сервісами. Вони дозволяють вирішувати широкий спектр завдань, від надання інформації до виконання складних функцій у реальному часі. Однак, є й обмеження, пов'язані з функціональними можливостями, масштабованістю та інтеграцією з іншими платформами, що можуть бути вирішені за допомогою додаткових технологій та ресурсів.

1.3 Огляд технологій для розробки Telegram-ботів (Python, Flask, Telegram API)

Розробка Telegram-ботів вимагає використання ряду технологій, які дозволяють створювати інтерактивні, функціональні та зручні у використанні боти. Оскільки Telegram надає API для створення ботів, важливими технологіями для реалізації є мови програмування, фреймворки та інтерфейси для взаємодії з API. Один з найбільш популярних та зручних варіантів для розробки Telegram-ботів — це використання Python, фреймворку Flask та Telegram API. У цьому розділі буде наведено детальний огляд цих технологій.

Python є однією з найпопулярніших мов програмування для створення Telegram-ботів завдяки своїй простоті, потужному екосистемі бібліотек та зручним засобам для інтеграції з API. Мова Python дозволяє швидко писати код, який буде підтримувати великі та складні проекти.

Серед переваг використання Python ключовою є простота синтаксису, оскільки Python є мовою високого рівня, що дозволяє швидко та ефективно писати код.

Наступною перевагою є багатий набір бібліотек. Для роботи з Telegram API існує спеціалізовані бібліотеки, наприклад, python-telegram-bot, які спрощують інтеграцію з Telegram.

Крім того Python має велику спільноту розробників, що дозволяє отримати допомогу та підтримку у разі виникнення проблем. Також Python дозволяє

створювати асинхронні боти, що дає змогу ефективно обробляти велику кількість запитів одночасно.

Популярні бібліотеки для роботи з Telegram в Python:

− python-telegram-bot: Одна з найбільш популярних бібліотек для створення ботів, яка надає зручний інтерфейс для роботи з Telegram API. Вона підтримує всі основні функції, такі як отримання повідомлень, відправка медіафайлів, клавіатур, обробка команд і багато іншого;

− telepot: Легка та швидка бібліотека для створення Telegram-ботів, яка дозволяє працювати з основними функціями бота;

− aiogram: Асинхронна бібліотека для роботи з Telegram API, яка дозволяє створювати більш масштабовані та продуктивні боти.

Далі розглянемо Flask — мікрофреймворк для веб-розробки. Це популярний мікрофреймворк для Python, який часто використовується для розробки веб-додатків і API, зокрема для створення веб-серверів, які можуть обробляти запити від Telegram. Flask дуже підходить для створення легких і швидких додатків, де важлива гнучкість і простота.

Серед ключових переваг використання Flask для розробки Telegram-ботів є легкість та простота: Flask — це мікрофреймворк, що означає мінімум налаштувань та конфігурацій, дозволяючи розробникам швидко зосередитись на реалізації бізнес-логіки. Також шнучкість Flask дозволяє розробникам мати повний контроль над компонентами додатка, без жорсткої структури.

Інтеграція з веб-сервісами Flask дає можливість швидко розробляти веб-сервіси та API, що дозволяє Telegram-боту обробляти вхідні запити за допомогою HTTP-запитів (наприклад, webhook).

Опишемо як Flask використовується для Telegram-ботів.

Перш за все, обробка webhook. Flask дозволяє налаштувати сервер для отримання HTTP-запитів від Telegram через webhook. Це важливо для забезпечення зворотного зв’язку з ботом в режимі реального часу.

По-друге, маршрутизація запитів. Flask дозволяє зручно налаштовувати маршрути для обробки різних запитів, таких як обробка команд або виконання певних дій після отримання повідомлень.

Розглянемо Telegram API — інтерфейс для створення ботів. Це набір інтерфейсів, який дозволяє взаємодіяти з серверами Telegram для виконання різних дій через бота. API надає всі необхідні функції для реалізації основних можливостей ботів: відправка і отримання повідомлень, обробка команд, робота з медіа, інтеграція з користувацькими інтерфейсами та багато іншого.

Опишемо основні можливості Telegram API. Перша це обробка повідомлень. API дозволяє отримувати та відправляти текстові та мультимедійні повідомлення (фото, відео, документи тощо). Наступною можливістю є обробка команд. Боти можуть реагувати на специфічні команди, що дозволяє створювати інтерактивні сценарії для користувачів.

Серед можливостей варто виділити інтеграцію з іншими сервісами. Telegram API дозволяє інтегрувати бота з іншими сервісами через API, обробляти запити користувачів та забезпечувати доступ до додаткових функцій.

Для зручності користувачів Telegram API надає можливість створювати інтерфейси з кнопками, що дозволяє зробити взаємодію з ботом більш інтуїтивно зрозумілою.

Telegram підтримує два основних способи взаємодії з серверами: webhook, який передає повідомлення безпосередньо на вказану URL-адресу, та long polling, що дозволяє постійно запитувати сервер для отримання нових повідомлень.

Опишемо як використовувати Telegram API для розробки бота. По-перше, підключення webhook для автоматичного отримання запитів від Telegram у реальному часі. Сервер Flask може слухати ці запити, обробляти їх і надсилати відповіді. По-друге, якщо ви не хочете налаштовувати сервер, можна використовувати long polling, де бот періодично звертається до серверів Telegram для отримання нових повідомлень.

Для розробки Telegram-ботів Python є чудовим вибором завдяки своїй простоті та потужним бібліотекам. Flask дозволяє створювати легкі та гнучкі веб-сервіси для обробки запитів від користувачів, а Telegram API забезпечує всі необхідні функції для взаємодії з ботом та створення ефективної автоматизації процесів. Використання цих технологій разом дозволяє створювати функціональні, масштабовані та прості у використанні Telegram-боти, що відповідають різноманітним потребам користувачів та бізнесу.

1.4 Вимоги до системи персонального помічника

Система персонального помічника (ПП) є елементом системи підтримки прийняття рішень (СППР) і повинна відповідати ряду вимог, які забезпечують її ефективність, зручність і безпеку використання. Розглянемо далі основні категорії вимог до системи персонального помічника.

Функціональні вимоги. До низ належать обробка запитів користувача, підтримка голосових команд (за потребою), інтеграція з іншими сервісами, визначення та збереження контексту, обробка запитів у режимі реального часу й прогнозування та рекомендації. Опишемо детальніше, у чому вони полягають.

Система повинна вміти обробляти текстові запити користувача, які можуть включати питання, команди, запити на отримання інформації або виконання певних дій (наприклад, створення задач, надання рекомендацій, відправка повідомлень). Для покращення взаємодії система повинна підтримувати голосові команди, дозволяючи користувачу взаємодіяти з помічником без використання клавіатури.

Персональний помічник має можливість інтегруватися з іншими онлайн-сервісами (календарі, поштові сервіси, соціальні мережі, бази даних, системи управління проектами та іншими інструментами). Помічник має здатність зберігати історію взаємодії з користувачем і використовувати контекст попередніх запитів для покращення якості та точності відповідей.

Система повинна бути здатна обробляти запити користувача в режимі реального часу, забезпечуючи миттєву відповідь на кожен запит. Помічник повинен надавати рекомендації та пропозиції на основі попередньої діяльності користувача, його інтересів і уподобань.

До нефункціональних вимог належать: продуктивність і масштабованість, інтуїтивно зрозумілий інтерфейс; доступність та сумісність; безпека та конфіденційність; надійність і доступність.

Відповідно до нефункціональних вимог система повинна працювати швидко, навіть коли кількість користувачів або запитів збільшується. Вона має бути масштабованою для обробки великої кількості запитів одночасно. Персональний помічник повинен мати простий і зрозумілий інтерфейс, що дозволяє користувачам легко налаштовувати та використовувати систему, навіть якщо у них немає технічних знань.

Система повинна бути доступною на різних платформах (мобільні пристрої, десктопи, веб-браузери) і підтримувати різноманітні операційні системи (Windows, macOS, Android, iOS). Всі особисті дані користувачів повинні бути захищені відповідно до найсучасніших стандартів безпеки. Система повинна включати механізми автентифікації, шифрування даних та контролю доступу до конфіденційної інформації. Помічник повинен бути доступний 24/7 з мінімальним часом простою. Система повинна бути стабільною і працювати без збоїв.

Наступною групою є інтерфейсні вимоги, до яких належать чат-інтерфейс, голосовий інтерфейс (за потребою), мультиканальність. Для зручності користувачів, особливо в рамках Telegram-бота, інтерфейс повинен бути текстовим і підтримувати основні функції чат-ботів, такі як клавіатури, кнопки, меню та можливість введення команд. Якщо система підтримує голосове введення, вона повинна розпізнавати голосові команди з високою точністю та мати можливість генерувати голосові відповіді.

Система повинна підтримувати взаємодію з користувачами через різні канали, включаючи текстові повідомлення, електронну пошту, голосові дзвінки та інші платформи (наприклад, через Telegram, WhatsApp, Slack).

До інтелектуальних вимог належать аналіз та обробка природної мови (NLP) та моделі машинного навчання.

Система повинна мати можливості для обробки запитів користувачів на природній мові, забезпечуючи точність та правильне розуміння контексту. Система повинна використовувати методи машинного навчання для постійного покращення своєї здатності до прогнозування, класифікації та рекомендацій, заснованих на аналізі даних користувачів та їх взаємодії з системою.

Серед технічних вимог виокремимо такі: платформа для розробки, інтеграція з базами даних, API для інтеграції. Відповідно до цих вимог система повинна бути розроблена з використанням ефективних технологій, таких як Python, фреймворк Flask, а також використовувати можливості Telegram API для створення бота. Для збереження інформації та історії взаємодії користувачів система повинна мати інтеграцію з реляційними або NoSQL базами даних.

Система повинна мати API для інтеграції з іншими зовнішніми сервісами, що дозволить розширити її функціональні можливості, наприклад, для отримання даних з інших платформ, баз даних або сервісів.

Також виокремимо вимоги до тестування: тестування на зручність використання, тестування на безпеку та тестування продуктивності. Необхідно проводити регулярні тести на зручність використання (usability testing), щоб переконатися в простоті взаємодії з користувачем і ефективності інтерфейсу.

Система повинна бути перевірена на наявність вразливостей у безпеці, особливо в аспектах захисту даних користувачів, автентифікації та доступу до конфіденційної інформації. Важливо провести тестування продуктивності для перевірки ефективності обробки великої кількості запитів та взаємодії з користувачами в умовах навантаження.

Також визначаються вимоги до документації, а саме: користувацька документація та технічна документація.

Необхідно розробити документацію для користувачів, яка містить інструкції з налаштування, використання та вирішення проблем, що можуть виникнути. Для розробників і адміністраторів повинна бути створена технічна документація, яка описує архітектуру системи, компоненти, API та інші важливі аспекти для підтримки та розвитку системи.

Отже, вимоги до системи персонального помічника охоплюють широкий спектр аспектів, від функціональності та інтерфейсу до безпеки та інтеграції з іншими сервісами. Важливою умовою успіху такої системи є забезпечення її ефективної роботи, зручності для користувачів та здатності адаптуватися до змінюваних умов і потреб.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ TELEGRAM-БОТІВ

2.1Вимоги до системи персонального помічника

Персональний помічник у вигляді Telegram-бота повинен забезпечувати користувачів актуальною інформацією про погоду та рекомендаціями, що стосуються здоров’я, на основі введених симптомів або загального стану. Бот має бути простим у використанні, надавати швидкі та зрозумілі відповіді, а також працювати цілодобово. Основною аудиторією бота є люди, які шукають зручний спосіб отримувати корисні поради або планувати свій день залежно від погодних умов.

Розглянемо функціональні вимоги. Бот повинен інтегруватися з API погоди для отримання актуальних даних про температуру, вологість, опади та інші метеорологічні показники в реальному часі. Крім того, він повинен використовувати модель штучного інтелекту для аналізу введених користувачем симптомів (наприклад, "болить горло" чи "висока температура") і пропонувати релевантні рекомендації (пити багато води, звернутися до лікаря тощо). Система повинна підтримувати багатомовність і надавати відповіді на основі контексту, щоб уникнути неоднозначності у взаємодії.

Опишемо технічні вимоги. Для ефективної роботи бота необхідно мати стабільне серверне середовище з інтеграцією бази знань про здоров’я та погодні умови. Відповіді мають генеруватися швидко (менше 2 секунд), навіть за значної кількості одночасних користувачів. Інтерфейс бота повинен бути інтуїтивно зрозумілим, підтримувати текстові повідомлення та кнопки для швидкого вибору дій. Для забезпечення конфіденційності даних користувача бот повинен відповідати сучасним стандартам захисту інформації.

Визначимо нефункціональні вимоги. Telegram-бот повинен бути масштабованим і підтримувати оновлення функціоналу без значних зупинок у

роботі. У разі виникнення проблем або некоректного вводу, бот повинен надавати корисні підказки чи пропонувати приклади можливих запитів. Також важливим є інтерактивність, щоб користувачі могли відчувати, що взаємодіють з персональним асистентом, а не просто зі стандартною програмою.

Запропонуємо можливості розширення. У майбутньому бот може інтегруватися з іншими сервісами, такими як системи розпізнавання голосу, календарі, медичні довідники або фітнес-трекери. Це дозволить значно розширити його функціонал, зробивши персонального помічника ще кориснішим для повсякденного життя користувачів.

2.2Вибір мови програмуванн

Для розробки Telegram-бота, який надаватиме відомості про погоду та рекомендації щодо здоров’я, Python є оптимальним вибором через кілька ключових переваг, які розглянемо далі.

По-перше, широка підтримка бібліотек для роботи з Telegram API. Python має популярну бібліотеку python-telegram-bot, яка забезпечує зручний інтерфейс для роботи з Telegram API. Ця бібліотека надає інструменти для обробки текстових повідомлень, кнопок, команд і інтерактивного спілкування з користувачем, що значно скорочує час розробки.

Важливим аспектом є інтеграція зі сторонніми сервісами. Python має багатий набір бібліотек для роботи з API інших сервісів. Для отримання даних про погоду можна використовувати бібліотеки на основі requests для запитів до OpenWeatherMap або інших погодних API. Для роботи з моделями штучного інтелекту є бібліотеки, такі як TensorFlow, PyTorch або Scikit-learn, що дозволяють інтегрувати нейронні мережі без складнощів.

Важливою є простота у використанні та підтримці. Python відомий своїм простим і зрозумілим синтаксисом, що полегшує написання, читання та підтримку коду. Завдяки цьому навіть менш досвідчені розробники можуть швидко впроваджувати зміни чи додавати нові функції.

Ключову роль відіграють потужні можливості обробки тексту. Python має інструменти для обробки тексту, такі як NLTK і spaCy, які можуть використовуватися для аналізу введених користувачем симптомів і формування відповідних рекомендацій. Це робить Python ідеальним для інтеграції алгоритмів обробки природної мови (NLP).

Також слід взяти до уваги масштабованість і доступність. Python легко масштабується для великих проєктів, а його кросплатформенна природа дозволяє запускати бота як локально, так і на серверах. Крім того, Python широко підтримується на хмарних платформах, таких як AWS, Google Cloud і Heroku, що спрощує деплой і підтримку бота.

Також Python має одну з найбільших спільнот розробників. Це означає, що існує безліч готових рішень, документації та форумів, де можна знайти відповіді на будь-які питання, що виникають під час розробки.

Хоча Python є основним кандидатом, також можна розглянути: JavaScript (Node.js) як варіант для інтерактивних веб-додатків і ботів, але потребує більше зусиль для інтеграції з моделями ШІ; Go підходить для високонавантажених систем, але менш популярний у контексті Telegram-ботів. Коротко розглянемо їх особливості також.

JavaScript – це мова програмування, яка спочатку була створена для роботи у веб-браузерах, але з появою Node.js стала універсальним інструментом для серверної розробки. Node.js – це середовище виконання JavaScript, що дозволяє виконувати код на сервері, використовуючи асинхронну обробку подій. Завдяки цьому Node.js добре підходить для обробки багатокористувацьких систем із високою пропускною здатністю.

Розглянемо переваги цієї мови. Node.js відзначається високою швидкодією завдяки асинхронній моделі роботи з подіями. Це дозволяє ефективно обробляти численні запити одночасно, зберігаючи продуктивність навіть у реальному часі. Додатково, екосистема Node.js, зокрема через NPM, забезпечує розробників багатим вибором бібліотек, таких як telegraf.js, які значно спрощують створення Telegram-ботів. Гнучкість JavaScript також є суттєвою перевагою, оскільки мову

можна використовувати як для серверної, так і для клієнтської частини, що дозволяє створювати додатки повного стеку з єдиним середовищем розробки.

Однак, Node.js має певні обмеження. Однопотокова архітектура може стати проблемою при виконанні ресурсомістких обчислень, створюючи затримки для інших запитів. Крім того, велика кількість асинхронного коду може ускладнювати його розуміння та налагодження, особливо для менш досвідчених розробників. Ще одним недоліком є відносно слабка підтримка інструментів для машинного навчання у порівнянні з Python, що може обмежити можливості інтеграції складних ШІ-моделей у проєкт.

Go – це компільована мова програмування, створена для побудови швидких і надійних систем. Вона характеризується простим синтаксисом і високою продуктивністю, що робить її популярною для створення серверних систем, мікросервісів і додатків із високими вимогами до продуктивності.

Go вирізняється високою продуктивністю завдяки компіляції в машинний код, що забезпечує швидке виконання програм. Вбудована підтримка горутин – легковагових потоків для роботи з конкурентними задачами – дозволяє ефективно використовувати ресурси системи, особливо для багатопотокових програм. Крім того, мова має сувору типізацію та мінімалістичний синтаксис, що сприяє написанню надійного коду з меншою кількістю помилок. Статична компіляція позбавляє потреби використовувати додаткові інтерпретатори або віртуальні машини, що спрощує розгортання програм у різних середовищах.

Незважаючи на переваги, Go має обмежену екосистему, особливо в контексті роботи з Telegram API, оскільки бібліотек для цього значно менше, ніж у Python чи JavaScript. Також мова менш популярна у сфері машинного навчання, тому інтеграція складних ШІ-моделей може бути непростою задачею. Ще однією складністю є крута крива навчання: новачкам, особливо тим, хто не знайомий зі статично типізованими мовами, може бути важко адаптуватися до особливостей Go, зокрема до роботи з конкурентністю та специфічною структурою програм.

JavaScript (Node.js) підходить для розробки Telegram-ботів, які потребують інтерактивності та обробки великої кількості запитів у реальному часі. Go ж

варто розглядати для проєктів із високими вимогами до продуктивності та надійності, хоча його екосистема менш розвинена для задач із ШІ. Обидві мови мають свої переваги та недоліки, і вибір залежить від специфіки проєкту та досвіду команди.

Отже, Python є найкращим вибором для створення Telegram-бота з функціями погоди та рекомендацій щодо здоров’я завдяки своїй простоті, потужним бібліотекам і можливостям інтеграції зі сторонніми сервісами. Це забезпечує швидку розробку, легку підтримку та зручне масштабування системи.

2.3Огляд Telegram API

Telegram API – це набір інструментів і методів, який дозволяє розробникам інтегрувати свої додатки з платформою Telegram. Завдяки цьому API можна створювати ботів, управляти каналами, групами, отримувати повідомлення, відправляти медіафайли, а також реалізовувати інтерактивну взаємодію з користувачами. Telegram API відзначається простотою використання та широкими можливостями, що робить його одним із найпопулярніших для створення чат-ботів [10].

Telegram API можна поділити на два основних типи: Telegram Bot API та Telegram Core API [21].

Telegram Bot API – орієнтований на розробку ботів, що взаємодіють із користувачами. Це RESTful API, який дозволяє розробникам легко створювати ботів для виконання різноманітних завдань, від автоматизації обробки повідомлень до складних інтеграцій.

Telegram Core API – призначений для створення клієнтських додатків і роботи з Telegram на рівні користувачів. Цей API надає доступ до всіх функцій Telegram, але вимагає значного рівня технічної експертизи та додаткових дозволів.

Розглянемо детальніше Telegram Bot API. Bot API забезпечує зручний спосіб створення ботів без необхідності прямого доступу до серверів Telegram.

Боти можуть обробляти повідомлення, реагувати на команди, взаємодіяти з іншими сервісами, надсилати медіафайли, керувати опитуваннями та навіть проводити платежі через інтеграцію з платіжними системами. Доступ до API надається через токен, який створюється за допомогою офіційного бота BotFather.

Розглянемо основні можливості Bot API [21]:

- робота з повідомленнями, а саме надсилання текстових повідомлень, зображень, відео, аудіо, документів та стікерів;
- реалізація інтерактивних кнопок та меню для зручності користувачів;
- боти можуть отримувати дані через вебхуки або опитування (polling);
- боти здатні обробляти та відправляти широкий спектр медіафайлів;
- можливість створення платіжних запитів через Telegram.

Розглянемо детальніше Telegram Core API. Цей API надає більш детальний доступ до функцій Telegram, включаючи взаємодію з особистими повідомленнями, групами та каналами. Core API більше підходить для створення сторонніх клієнтів Telegram або складних інтеграцій. Використання цього API вимагає реєстрації додатка та отримання дозволів від Telegram.

Переваги Telegram API наведено нижче:

- простота використання завдяки добре задокументованому інтерфейсу, який легко інтегрувати в додатки;
- підтримка широкого спектру функцій – від простих текстових повідомлень до інтеграції зі сторонніми сервісами;
- Telegram API є повністю безкоштовним, що відкриває можливості для розробників різного рівня.

Telegram API має ліміти на кількість запитів, щоб забезпечити стабільність сервісу. Наприклад, боти можуть обробляти не більше ніж 30 повідомлень на секунду для одного токена. Для уникнення блокувань необхідно дотримуватися цих лімітів і оптимізувати код.

Telegram API – це потужний і доступний інструмент для створення сучасних інтерактивних ботів та клієнтів, який залишається популярним завдяки своїй гнучкості, можливостям і простоті інтеграції.

2.4Інтеграція з бібліотеками та платформами ШІ

Інтеграція Telegram-бота з бібліотеками та платформами штучного інтелекту (ШІ) дозволяє розширити функціональність бота та створити складні системи рекомендацій, обробки тексту, аналізу даних та багато іншого. Це відкриває широкий спектр можливостей, таких як автоматичне генерування відповідей, персоналізація рекомендацій, розпізнавання зображень та голосу, а також впровадження моделей машинного навчання.

Розглянемо популярні бібліотеки ШІ для інтеграції [18, 19].

TensorFlow — це популярна бібліотека з відкритим кодом для машинного навчання, створена компанією Google. Вона використовується для побудови, тренування та розгортання моделей машинного навчання, включаючи нейронні мережі. TensorFlow підтримує широкий спектр завдань, таких як класифікація, регресія, розпізнавання зображень, обробка природної мови (NLP) і багато інших.

Розглянемо особливості TensorFlow. Перш за все це гнучкість у розробці моделей. TensorFlow дозволяє створювати моделі як високого, так і низького рівня складності. Завдяки API високого рівня (наприклад, Keras) користувачі можуть швидко створювати прототипи, а функції низького рівня дають змогу налаштовувати кожен аспект моделі [18, 19].

Важливим аспектом є підтримка багатьох платформ. Моделі TensorFlow можна запускати на різних платформах — від серверів і настільних комп'ютерів до мобільних пристроїв і вбудованих систем. Це забезпечується такими інструментами, як TensorFlow Lite та TensorFlow.js.

TensorFlow підтримує тренування моделей на кількох графічних процесорах (GPU) або навіть у кластері серверів, що значно прискорює обробку

великих наборів даних. TensorFlow легко інтегрується з хмарними платформами, такими як Google Cloud, AWS і Azure, що дозволяє виконувати складні обчислення на потужних серверах.

Визначимо основні переваги TensorFlow:

1. Потужна екосистема. TensorFlow має розвинуту екосистему інструментів, включаючи TensorBoard для візуалізації навчання моделі, TensorFlow Hub для повторного використання попередньо натренованих моделей та TensorFlow Extended (TFX) для управління повним життєвим циклом моделі.

2. Активна спільнота. Завдяки активній спільноті розробників та великій кількості навчальних матеріалів, TensorFlow ідеально підходить як для новачків, так і для досвідчених користувачів.

3. Підтримка різних мов програмування. Хоча основною мовою TensorFlow є Python, бібліотека також має підтримку для C++, JavaScript (TensorFlow.js) і Java, що забезпечує кросплатформену інтеграцію.

Проте TensorFlow має такі недоліки:

1. Крута крива навчання. Хоча Keras спрощує розробку моделей, низькорівневі функції TensorFlow можуть бути складними для розуміння, особливо для новачків.

2. Велика кількість абстракцій. Деякі частини бібліотеки можуть здаватися надмірно складними через велику кількість рівнів абстракції.

3. Ресурсоємність. TensorFlow вимагає значних апаратних ресурсів, особливо для великих моделей, що може бути проблемою для малих проєктів без доступу до потужних серверів або GPU.

TensorFlow може бути інтегрований у Telegram-боти для виконання завдань, таких як: розпізнавання зображень, які користувач надсилає боту; генерація відповідей за допомогою NLP-моделей; використання рекомендативних систем для надання персоналізованих порад.

Для цього можна розробити модель у TensorFlow, розгорнути її на сервері та інтегрувати з Telegram API через вебхуки чи опитування. Це дозволяє ботові

виконувати обчислення в реальному часі та значно покращує його функціональність.

PyTorch – це ще одна популярна бібліотека для машинного навчання та глибокого навчання з відкритим кодом, розроблена компанією Facebook. Вона широко використовується для створення нейронних мереж, глибинного навчання та обробки природної мови (NLP). PyTorch пропонує гнучкий та динамічний підхід до моделювання, що робить її однією з найбільш вибіркових бібліотек серед дослідників і розробників [18, 19].

Опишемо особливості PyTorch. Однією з основних відмінностей PyTorch від TensorFlow є використання динамічних графів. Це означає, що графи обчислень створюються "на льоту" під час виконання, що дозволяє швидко змінювати модель, тестувати нові ідеї та модифікувати архітектуру без необхідності переписувати код. Це робить PyTorch дуже зручним для досліджень і експериментів. PyTorch надає користувачам можливість працювати з абстракціями високого рівня (наприклад, для побудови нейронних мереж через модулі, як torch.nn) та з більш низьким рівнем абстракції, що дозволяє мати повний контроль над процесом тренування моделі. Це також полегшує навчання та підтримку моделі. PyTorch має вбудовану систему автоматичного обчислення похідних, що дозволяє автоматично обчислювати градієнти для оптимізації моделей машинного навчання. Це важлива функція для тренування нейронних мереж.

Як і TensorFlow, PyTorch підтримує обчислення на графічних процесорах (GPU), що дозволяє значно прискорити навчання моделей, особливо при роботі з великими наборами даних. PyTorch містить потужні модулі, такі як torchvision для комп'ютерного зору, torchaudio для обробки аудіо та torchtext для роботи з текстами. Це дозволяє значно спростити процес створення моделей для специфічних задач.

Переваги PyTorch наведено нижче:

1. PyTorch має інтуїтивно зрозумілий інтерфейс і не потребує складних налаштувань для початку роботи. Це робить його відмінним вибором для новачків та експериментаторів.

2. PyTorch користується великою популярністю серед дослідників і має активну спільноту, яка створює численні ресурси, бібліотеки та документацію. Бібліотека постійно вдосконалюється, і нові інструменти з'являються дуже часто.

3. Динамічні графи дозволяють легко проводити наукові експерименти з модифікацією моделей, що робить PyTorch привабливою для науковців і розробників, які працюють у галузі глибокого навчання та штучного інтелекту.

4. PyTorch підтримує взаємодію з іншими популярними бібліотеками для машинного навчання та наукових обчислень, такими як NumPy, SciPy і Cython, що робить його дуже гнучким для інтеграції у різні робочі процеси.

Недоліки PyTorch:

1. Хоча PyTorch дуже потужний для досліджень, раніше він вважався менш готовим для розгортання у промислових умовах порівняно з TensorFlow. Однак ситуація змінилася, і тепер PyTorch має інструменти для розгортання моделей, такі як TorchServe і TorchScript, що дозволяє ефективно працювати у виробничих середовищах.

2. Хоча PyTorch має потужну екосистему для глибинного навчання, у порівнянні з TensorFlow, деякі інші галузі, наприклад, робота з мобільними або веб-додатками, можуть бути не так добре підтримувані.

PyTorch може бути інтегрований у Telegram-боти для широкого спектру завдань, таких як: класифікація текстів або повідомлень для створення чат-ботів, що надають рекомендації або автоматично відповідають на запити; розпізнавання зображень, надісланих користувачами, для автоматичного визначення вмісту фотографій; розпізнавання голосу або синтез тексту на основі голосових повідомлень користувачів; створення систем рекомендацій, що надають персоналізовані поради на основі аналізу попередніх взаємодій з користувачем.

Інтеграція PyTorch в Telegram-бота може здійснюватися через API для виклику моделі, натренованої на сервері. Це дозволяє виконувати обчислення та надавати відповіді в реальному часі, покращуючи взаємодію з користувачами.

spaCy — це одна з найпопулярніших бібліотек для обробки природної мови (NLP) в Python. Вона була розроблена з акцентом на ефективність, швидкість та легкість інтеграції в реальні додатки. spaCy дозволяє розв'язувати широкий спектр завдань NLP, таких як токенізація, лемматизація, частотний аналіз, розпізнавання сутностей (NER), класифікація текстів та багато іншого [18, 19].

Однією з головних переваг spaCy є її висока швидкість обробки тексту, що робить її ідеальним вибором для застосувань, де необхідна обробка великих обсягів даних в реальному часі. Бібліотека оптимізована для швидкого виконання операцій, таких як токенізація, лемматизація та розпізнавання сутностей, що дозволяє ефективно працювати з великими даними, зберігаючи при цьому високу продуктивність.

Іншою важливою особливістю spaCy є наявність вбудованих попередньо натренованих моделей для численних мов, що дозволяє легко використовувати її для широкого спектру завдань NLP, таких як розпізнавання сутностей (NER), аналіз залежностей і лексичний аналіз. Завдяки простоті інтеграції з іншими бібліотеками машинного навчання, такими як TensorFlow і PyTorch, spaCy також дає можливість будувати та адаптувати власні моделі для специфічних завдань. Це робить її універсальним інструментом для побудови продуктивних та масштабованих систем обробки природної мови.

Основними перевагами spaCy є її висока швидкість та ефективність при обробці великих обсягів тексту, що робить її відмінним вибором для реальних проєктів з вимогами до швидкості виконання. Бібліотека підтримує широкий спектр завдань обробки природної мови, таких як токенізація, частотний аналіз, аналіз залежностей та розпізнавання сутностей, що дозволяє використовувати її для багатьох типових завдань NLP. Додатково, spaCy забезпечує зручну інтеграцію з іншими бібліотеками, такими як PyTorch і TensorFlow, що дозволяє застосовувати її разом з більш складними моделями для специфічних завдань.

Однак spaCy має й деякі недоліки. Зокрема, її гнучкість може бути меншою порівняно з іншими бібліотеками, такими як NLTK, яка надає більш широкий набір інструментів для спеціалізованого аналізу тексту. Крім того, хоча spaCy охоплює багато основних завдань NLP, вона не має всіх методів аналізу морфології та синтаксису, які можна знайти в більш спеціалізованих інструментах або бібліотеках, таких як NLTK. Це може бути обмеженням для деяких специфічних завдань, що потребують більш глибокого аналізу тексту.

NLTK (Natural Language Toolkit) — це одна з найстаріших і найбільш використовуваних бібліотек для обробки природної мови в Python. Вона забезпечує розширений набір інструментів для роботи з текстовими даними, таких як токенізація, морфологічний аналіз, часткове позначення частин мови (POS), лемматизація, розпізнавання сутностей, класифікація та багато іншого [18, 19].

Однією з головних переваг NLTK є її величезна кількість інструментів для аналізу тексту, що дозволяє користувачам виконувати різноманітні завдання в сфері обробки природної мови. Вона включає моделі для морфологічного аналізу, синтаксичного аналізу, класифікації текстів, створення корпусів і статистичного аналізу. Цей широкий набір інструментів робить NLTK потужним інструментом для досліджень у галузі NLP, надаючи велику кількість ресурсів для науковців і розробників.

NLTK надає великий набір корпусів текстів і мовних моделей для різних мов, що дозволяє користувачам тренувати і тестувати моделі для різних завдань NLP. Ці корпуси включають матеріали для навчання моделей, таких як визначення частин мови, класифікація текстів та інші завдання. Такий набір ресурсів значно полегшує розробку моделей і їх тестування, особливо для початківців і науковців, які працюють з мовними даними.

Однією з переваг NLTK є її висока гнучкість. Бібліотека дозволяє будувати та налаштовувати власні інструменти обробки тексту, що дає можливість користувачам адаптувати бібліотеку до своїх специфічних потреб. Завдяки простому інтерфейсу, користувачі можуть інтегрувати нові методи і підходи для

вирішення специфічних завдань, що робить NLTK дуже потужним інструментом для досліджень і розробки нових методів обробки текстів.

NLTK також підтримує широкий спектр статистичних методів для обробки тексту, таких як наївні баєсові класифікатори, моделі на основі максимального правдоподібності та інші. Це дає можливість застосовувати потужні статистичні підходи для вирішення завдань класифікації, прогнозування та інших завдань NLP. Статистичні методи дають гнучкість у виборі підходу для конкретної задачі та дозволяють отримати високі результати у різних сферах.

Однією з основних переваг є широкий набір інструментів і корпусів, що надаються бібліотекою, а також велика кількість попередньо натренованих моделей для різних мов і завдань. Завдяки цьому NLTK є чудовим вибором для досліджень у сфері обробки текстів. Гнучкість і розширюваність бібліотеки дозволяють налаштовувати алгоритми та створювати власні рішення для специфічних завдань. Окрім того, наявність великої кількості навчальних матеріалів і документації робить її ідеальною для навчання та для тих, хто тільки починає працювати з NLP.

Одним з головних недоліків є її швидкість. NLTK не так оптимізована для обробки великих обсягів тексту в реальному часі, як інші бібліотеки, наприклад, spaCy. Це може стати обмеженням для проектів, де потрібна висока швидкість обробки даних. Крім того, хоча NLTK є дуже потужним інструментом для наукових досліджень, її менш зручне використання у реальних промислових проєктах, де критичними є швидкість і ефективність виконання задач.

OpenAI GPT – це потужна модель для обробки природної мови, розроблена компанією OpenAI, що використовується для генерації тексту, розв'язання задач, перекладу, аналізу тексту та інших завдань NLP. Бібліотека OpenAI GPT надає API для інтеграції з різними програмами та сервісами, включаючи чат-боти, аналітичні інструменти та персоналізовані помічники. Основною перевагою є здатність моделі генерувати тексти, що виглядають як написані людиною, а також адаптуватися до різних контекстів завдяки великому обсягу попереднього

навчання. Недоліком є висока вартість використання API, що може бути обмеженням для деяких проєктів [18, 19].

Hugging Face – це популярна платформа та бібліотека, яка надає доступ до великої кількості попередньо натренованих моделей для різних завдань у галузі машинного навчання, таких як NLP, комп'ютерне зір, класифікація тексту та ін. Hugging Face особливо популярна завдяки своїй бібліотеці Transformers, яка підтримує безліч сучасних моделей, включаючи BERT, GPT-2, T5 та інші, що дозволяє користувачам легко інтегрувати складні моделі в свої програми. Перевагою є широкий набір інструментів та моделей з відкритим кодом, що робить бібліотеку доступною для дослідників та розробників. Недоліком є потреба в потужних ресурсах для запуску великих моделей, що може бути обмеженням для малих проєктів [18, 19].

Інтеграція з ШІ значно покращує користувацький досвід завдяки персоналізації, ефективності та інтерактивності. Моделі ШІ дозволяють створювати рекомендації, адаптовані до індивідуальних потреб і поведінки користувачів, що робить взаємодію більш релевантною. Автоматизація рутинних завдань, таких як обробка запитів та відповідей, підвищує ефективність і знижує час відповіді. Використання технологій обробки природної мови (NLP) та генеративних моделей дозволяє забезпечити більш природну, гнучку та продуктивну взаємодію між ботом та користувачем.

Розглянемо деякі технічні аспекти інтеграції телеграм-ботів з ШІ.

Для інтеграції Telegram-бота з ШІ можна використовувати два основних підходи: вебхуки та опитування. Вебхуки дозволяють отримувати повідомлення від Telegram API в реальному часі, що дає можливість оперативно обробляти запити користувачів і передавати їх на сервер, де виконуються моделі ШІ. Опитування, в свою чергу, передбачає періодичне опитування API Telegram для отримання нових повідомлень, що може бути корисним для менш чутливих до часу завдань, але менш ефективним для забезпечення миттєвої реакції.

Багато популярних платформ для ШІ, таких як OpenAI, Hugging Face, або TensorFlow, надають REST API, що дозволяє зручно інтегрувати моделі

машинного навчання в Telegram-бота. Через API можна відправляти запити для обробки текстів, отримувати рекомендації, прогнози та інші відповіді від ШІ, що дозволяє боту виконувати завдання, від простих відповідей до складних аналітичних процесів. REST API забезпечує стандартний спосіб взаємодії з різними платформами, що робить процес інтеграції гнучким і масштабованим.

Використання хмарних сервісів (таких як AWS, Google Cloud або Azure) для розгортання моделей ШІ є ключовим аспектом інтеграції, оскільки вони забезпечують обчислювальні потужності для виконання складних обчислень. Хмари дозволяють ефективно масштабувати ресурси, що особливо важливо для обробки великих обсягів даних або виконання обчислень у реальному часі. Завдяки хмарній інфраструктурі забезпечується не тільки доступність моделей ШІ, але й безпека даних, що є важливим для інтеграції в комерційні та чутливі до безпеки додатки.

Інтеграція з ШІ перетворює Telegram-ботів із простих інструментів взаємодії на потужні рішення для автоматизації задач. Завдяки таким технологіям, як вебхуки, REST API та хмарна інфраструктура, боти стають здатними не лише відповідати на запити, але й активно допомагати користувачам, надаючи персоналізовані рекомендації, прогнози, аналітику та інші складні сервіси, що значно покращує досвід взаємодії з ботами.

2.5Використання нейромереж та методів машинного навчання для формування рекомедацій

Використання нейромереж для формування рекомендацій є однією з найпопулярніших та ефективних практик в індустрії, оскільки нейромережі здатні аналізувати великі обсяги даних і виявляти складні патерни, які важко помітити традиційними методами. Такі системи здатні створювати персоналізовані рекомендації на основі історії взаємодії користувача з продуктами, послугами чи контентом. Наприклад, у випадку з інтернет-магазинами, нейромережі можуть аналізувати попередні покупки, перегляди

товарів, а також інші поведінкові фактори, щоб запропонувати користувачу найбільш відповідні товари.

Основною перевагою нейромереж для рекомендацій є здатність обробляти великий обсяг даних без попереднього програмування спеціальних правил. Нейромережі, такі як згорткові або рекурентні, можуть обробляти як структуровану, так і неструктуровану інформацію (наприклад, текст або зображення), що дає змогу системі робити точні прогнози навіть з дуже різноманітними даними. У випадку з Telegram-ботами для рекомендацій з області здоров’я, нейромережа може аналізувати не лише медичні записи користувача, але й поведінкові патерни, на основі яких пропонувати персоналізовані поради.

Існують різні підходи до використання нейромереж для рекомендацій. Один із найпоширеніших методів – це колаборативна фільтрація, яка ґрунтується на припущенні, що схожі користувачі мають схожі переваги. Цей метод використовує дані про взаємодії користувачів з системою для побудови рекомендацій. Водночас, контентна фільтрація використовує властивості об’єктів (наприклад, опис товарів або характеристик продуктів) для створення рекомендацій, орієнтуючись на схожість між контентом, який користувач уже оцінив, та тим, який йому може сподобатися.

Для покращення якості рекомендацій можна також використовувати гібридні підходи, які комбінують кілька методів. Наприклад, в системах рекомендацій, що використовують нейромережі, поєднуються колаборативна фільтрація та контентний аналіз, що дозволяє отримати більш точні та персоналізовані рекомендації, враховуючи як уподобання конкретного користувача, так і специфіку предметної області. Гібридні моделі можуть також враховувати фактори, як-от контекст, у якому користувач здійснює взаємодію з системою.

Нейромережі також можуть адаптувати свої рекомендації на основі зворотного зв'язку від користувачів. Наприклад, якщо користувач відмовляється від певної рекомендації, система може швидко адаптувати свої прогнози, щоб

врахувати нові вподобання користувача. Це забезпечує динамічне налаштування рекомендацій, що є критично важливим для створення адаптивних і чутливих до змін систем взаємодії. Таким чином, використання нейромереж для рекомендацій є потужним інструментом, який може значно покращити досвід користувачів, пропонуючи їм найкращі та найбільш релевантні варіанти [11-15].

Для створення рекомендацій у Telegram-ботах найчастіше використовуються різні типи нейромереж, які ми розглянемо нижче.

Багатошарові перцептрони (MLP) — використовуються для побудови моделей, які можуть прогнозувати або класифікувати рекомендації на основі вхідних даних. Багатошарові перцептрони добре справляються з простими завданнями, де є потреба в аналізі числових ознак або категоріальних даних, таких як оцінки товарів чи послуг, вибір користувача тощо [11].

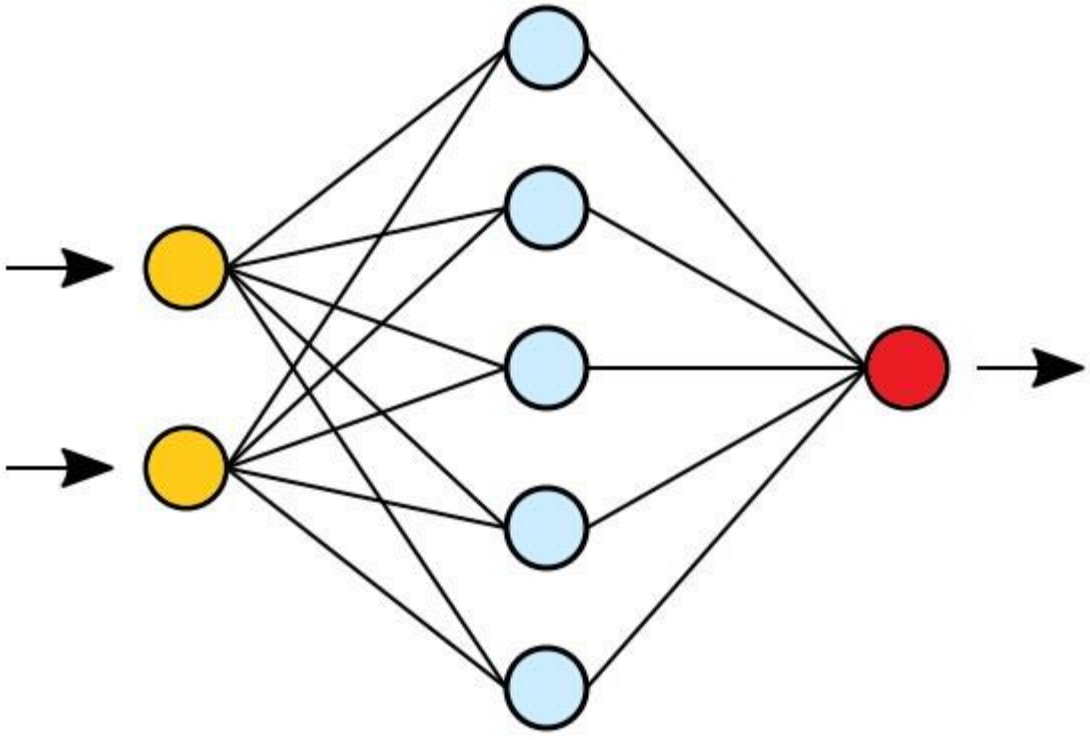

Рисунок 2.1 – Приклад багатошарового перцептрона

Багатошарові перцептрони (MLP) — це тип нейронної мережі, що складається з кількох шарів нейронів, включаючи вхідний, прихований і

вихідний шари. Кожен нейрон у мережі зв'язаний з нейронами в наступному шарі через ваги, які визначають важливість кожного входу. Мережа навчається шляхом корекції цих ваг, щоб мінімізувати різницю між передбаченими та фактичними виходами за допомогою алгоритму зворотного розповсюдження помилки.

У процесі навчання MLP приймає вхідні дані, пропускає їх через кілька прихованих шарів, де вони обробляються за допомогою нелінійних активаційних функцій, і на виході генерує прогноз. Такі моделі добре підходять для задач, де потрібно знайти складні закономірності між вхідними даними, наприклад, для класифікації або регресії. Багатошарові перцептрони є основою для багатьох сучасних моделей, таких як глибокі нейронні мережі.

Рекурентні нейромережі (RNN) — підходять для задач, де важлива послідовність або історія дій користувача, наприклад, при рекомендованих системах, що аналізують історію взаємодії з ботом. RNN можуть обробляти послідовні дані (наприклад, попередні запити користувача) та надавати рекомендації, базуючись на цьому контексті [12].

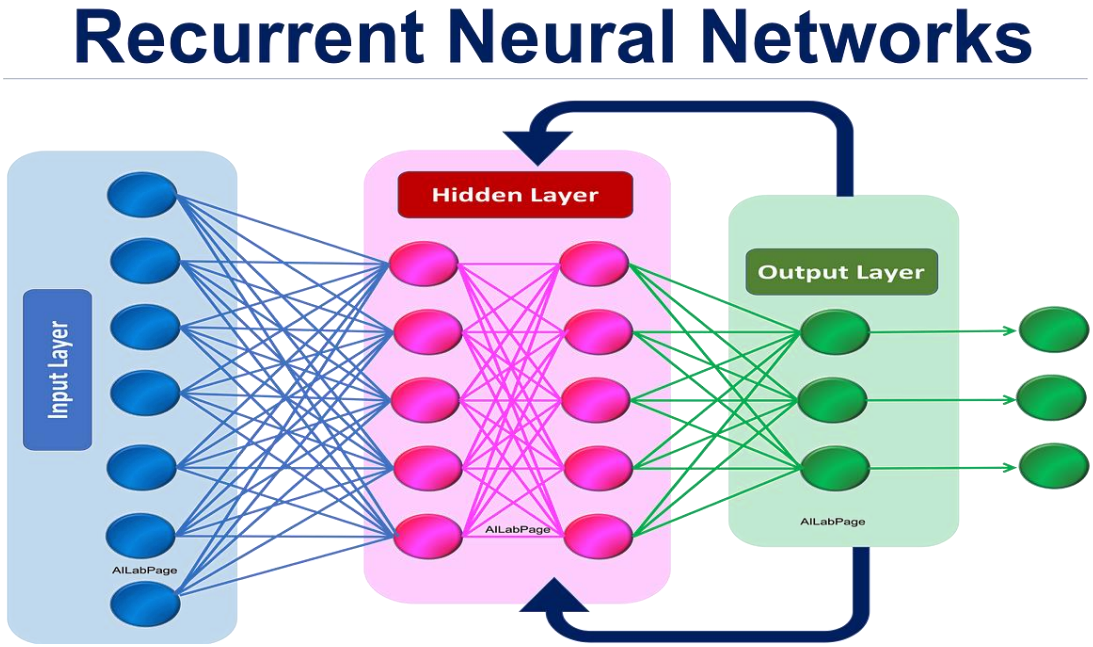

Рисунок 2.2 – Приклад рекурентної нейромережі

Рекурентна нейронна мережа (RNN) — це тип нейронної мережі, яка спеціально призначена для обробки послідовних даних, таких як текст, часоряди або відео. Основною особливістю RNN є наявність циклічних з'єднань між нейронами, що дозволяє їй зберігати інформацію про попередні стани. Це дає змогу мережі "пам'ятати" контекст і використовувати цю інформацію для передбачення наступних елементів у послідовності.

Під час обробки даних, мережа бере на вхід поточний елемент послідовності і, окрім цього, також враховує стан, який було отримано на попередньому етапі. Це дозволяє моделі адаптуватися до змін у часі та забезпечувати більш точні прогнози, враховуючи контекст. Однак, стандартні RNN мають проблеми з довготривалою пам'яттю через ефект затухаючого градієнта, що обмежує здатність мережі запам'ятовувати довготривалі залежності.

Щоб подолати ці обмеження, були розроблені вдосконалені версії RNN, такі як Long Short-Term Memory (LSTM) та Gated Recurrent Units (GRU). Ці моделі мають спеціальні механізми для ефективного зберігання та забування інформації, що значно покращує їх здатність до роботи з довгими послідовностями, зокрема в задачах, таких як машинний переклад, розпізнавання мови або створення тексту.

Трансформери — архітектури трансформерів, такі як BERT чи GPT, використовуються для створення рекомендованих систем на основі контекстуальних взаємодій. Ці моделі ефективно обробляють великі обсяги текстових даних і можуть виводити релевантні рекомендації, розуміючи контекст запитів. Наприклад, трансформери можуть бути корисними для створення рекомендацій на основі текстових запитів або взаємодії з користувачем [13].

Трансформери — це архітектура нейронних мереж, яка використовує механізм уваги (attention) для обробки послідовностей даних. Вони дозволяють моделі зосереджуватися на різних частинах вхідних даних, незалежно від їх

позиції в послідовності, що значно підвищує ефективність при обробці довгих послідовностей.

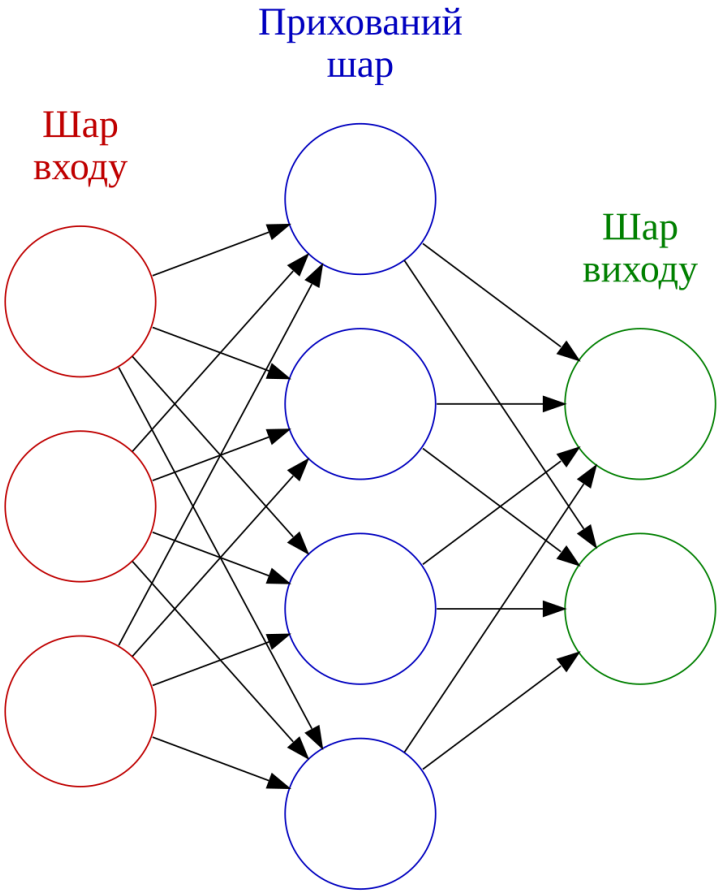

Рисунок 2.3 – Трансформер

Трансформери не залежать від рекурентних зв'язків, як RNN, і можуть бути оброблені паралельно, що значно прискорює навчання. Це робить їх дуже популярними для задач, пов'язаних з обробкою тексту, наприклад, у машинному перекладі, генерації тексту та інших задачах NLP.

Колаборативні фільтри (Collaborative Filtering) — хоча це не є нейромережею, колаборативні фільтри є популярним підходом у створенні рекомендацій, і можуть бути інтегровані в Telegram-ботів. Цей метод дозволяє створювати рекомендації на основі поведінки інших користувачів, що схожі на поточного користувача. Це може бути реалізовано через моделі машинного навчання, включаючи нейромережі для прогнозування переваг користувача [16].

Колаборативні фільтри (Collaborative Filtering) — це метод рекомендацій, який базується на ідеї, що користувачі, які виявляють подібні вподобання в минулому, ймовірно, будуть мати схожі вподобання в майбутньому. Існує два основних підходи до колаборативних фільтрів:

1. Метод на основі користувачів (User-based Collaborative Filtering) — цей підхід передбачає, що користувачі, які взаємодіяли з подібними елементами (наприклад, оцінювали фільми однаково), ймовірно, будуть рекомендувати подібні елементи іншим користувачам з подібними вподобаннями. Наприклад, якщо користувач A і користувач B мають схожі оцінки для певних фільмів, то для користувача A можуть бути рекомендовані фільми, які сподобалися користувачу B.
2. Метод на основі елементів (Item-based Collaborative Filtering) — цей підхід орієнтований на схожість між елементами. Рекомендації будуються на основі того, які елементи (наприклад, фільми) були спільно оцінені або взаємодіяли з іншими елементами. Якщо користувачі, які взаємодіють з фільмом X, також часто взаємодіють з фільмом Y, то цей фільм може бути рекомендований користувачам, які ще не взаємодіяли з ним.

Колаборативні фільтри мають кілька переваг, зокрема, вони не потребують знання контексту елементів (наприклад, теми фільмів) і здатні генерувати рекомендації, засновані на поведінці користувачів. Однак, вони мають і свої недоліки, такі як проблема "холодного старту" (коли немає достатньо даних про нових користувачів або елементи) і схильність до "вибуху вимірів" при великій кількості користувачів та елементів.

Автокодери (Autoencoders) — ці моделі використовуються для зниження вимірності даних та виявлення прихованих патернів у великих наборах даних. Автокодери використовуються в рекомендаційних системах для побудови моделей, які можуть зберігати важливі характеристики даних користувача та на основі цього генерувати персоналізовані рекомендації [11-15].

Автокодери (Autoencoders) — це тип нейронних мереж, які використовуються для навчання ефективного представлення (кодування)

вхідних даних у стислому вигляді. Вони складаються з двох основних частин: кодувальника (encoder) і декодувальника (decoder). Кодувальник приймає вхідні дані і перетворює їх у зменшену, більш компактну репрезентацію, яка називається кодовим вектором. Декодувальник, у свою чергу, відновлює оригінальні дані з цієї стислої репрезентації.

Основною метою автокодерів є мінімізація різниці між вхідними і відновленими даними. Це досягається через оптимізацію параметрів мережі таким чином, щоб вектор, отриманий після кодування, містив лише найбільш важливу інформацію для відновлення вхідних даних. Як результат, автокодери можуть бути використані для задач стиснення даних, видалення шуму з даних або навчання на основі неповних даних.

Автокодери знаходять застосування в багатьох сферах, зокрема в обробці зображень, звуків і текстів. Вони часто використовуються для зменшення розмірності даних або для створення нових, більш ефективних представлень даних. Однак, автокодери можуть мати обмеження при обробці дуже складних або різноманітних даних, оскільки їх здатність до стиснення інформації може бути обмежена особливостями архітектури мережі.

Всі ці нейромережі і методи можуть бути використані для створення інтелектуальних рекомендацій у Telegram-ботах, допомагаючи адаптувати їхні відповіді та пропозиції до інтересів та поведінки користувачів.

Для формування системи рекомендацій використовуються різні методи машинного навчання, які можуть бути поділені на три основні категорії [23-24]:

1. Фільтрація за колаборацією (Collaborative Filtering). Це метод, який аналізує взаємодії користувачів із предметами (наприклад, продуктами або фільмами) і рекомендує нові елементи на основі схожих уподобань інших користувачів.

Варіантом такої фільтрації є фільтрація за користувачами (User-based Collaborative Filtering). Рекомендується те, що подобається користувачам, схожим на вас. Це передбачає, що якщо два користувачі оцінили схожі елементи високо, вони, ймовірно, оцінять схожі елементи.

Інший варіант – фільтрація за елементами (Item-based Collaborative Filtering). Рекомендуються елементи, схожі на ті, що вже сподобалися користувачу. Наприклад, якщо користувач оцінив фільм високо, йому можуть порекомендувати схожі фільми.

До методів машинного навчання у такому випадку належать:

– алгоритм k-найближчих сусідів (k-NN). Використовується для пошуку схожих користувачів або предметів;

– матриця факторизації. Використовується для прогнозування відсутніх оцінок на основі оцінок інших користувачів. Популярними є методи, такі як SVD (Singular Value Decomposition) та ALS (Alternating Least Squares).

2. Фільтрація на основі контенту (Content-Based Filtering)

Цей метод рекомендує елементи на основі їх характеристик, таких як текстова інформація, жанр, характеристики продукту або інші метадані. Користувачеві рекомендуються товари або інші елементи, які мають схожі характеристики з тими, що йому сподобалися раніше. Використовується інформація про предмети (наприклад, жанри фільмів, опис книг або ключові слова в текстах), щоб порівняти їх із вподобаннями користувача.

Відповідні методи машинного навчання:

– класифікація (наприклад, на основі алгоритмів Naive Bayes, SVM). Для створення моделей, що визначають схожість між елементами;

– кластеризація (наприклад, K-means): Для групування схожих предметів в контексті особливостей;

– векторизація тексту (TF-IDF, Word2Vec, GloVe): Для аналізу та порівняння текстової інформації.

3. Гібридні методи. Це комбінація підходів фільтрації за колаборацією і на основі контенту, що дозволяє використовувати переваги обох методів і покращити якість рекомендацій.

Методи машинного навчання:

– комбінація фільтрації за колаборацією і контенту. Наприклад, можна комбінувати відомості про користувачів і предмети для створення більш точних рекомендацій;

– гібридні моделі з нейронними мережами. Використання глибокого навчання для комбінування різних типів даних, наприклад, Deep Collaborative Filtering.

4. Моделі на основі глибокого навчання (Deep Learning)

Глибокі нейронні мережі стають все популярнішими в системах рекомендацій, особливо в обробці великих обсягів даних або складних завданнях.

Методи машинного навчання:

– автокодери (Autoencoders). Використовуються для зменшення розмірності та пошуку схожих предметів або користувачів;

– генеративні моделі (GANs). Використовуються для створення нових рекомендаційних елементів на основі навчання на великих наборах даних;

– глибокі нейронні мережі. Моделі на основі нейронних мереж, такі як Recurrent Neural Networks (RNN) або Convolutional Neural Networks (CNN), можуть використовуватися для аналізу часових рядів або зображень.

5. Рекомендації на основі контексту (Context-Aware Recommender Systems). Цей підхід враховує додаткові контекстуальні фактори, наприклад, час доби, місце, пристрій або будь-які інші змінні, які можуть вплинути на рекомендації.

Методи машинного навчання:

– рекомендації на основі часового контексту. Використання моделей, таких як Recurrent Neural Networks (RNN), для обробки інформації про зміни в поведінці користувача в часі;

– персоналізовані моделі. Використовуються додаткові фактори, як поведінка в реальному часі, для створення точніших рекомендацій.

Отже, для створення систем рекомендацій використовуються різноманітні методи машинного навчання, в тому числі: фільтрація за колаборацією

(Collaborative Filtering); фільтрація на основі контенту (Content-Based Filtering); гібридні методи: глибоке навчання та нейронні мережі; контекстуальні рекомендації. Вибір методу залежить від специфіки задачі та доступних даних.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ТЕЛЕГРАМ-БОТА “AI ADVICER”

3.1Аналіз вимог до створення телеграм-бота “AI Advicer”

Розглянемо основні вимоги до створення телеграм-бота “AI Advicer”:

1Функціональні вимоги. Телеграм-бот "AI Advicer" має надавати рекомендації на основі введених користувачем запитів, що включають запити на основі стану здоров'я, погоди та особистих уподобань. Для цього бот повинен бути здатним обробляти текстові повідомлення, інтерпретувати запити та генерувати персоналізовані рекомендації. Бот повинен здійснювати інтеграцію з такими платформами, як погода API для надання актуальної інформації про погоду та моделями ШІ для рекомендацій по здоров'ю та іншими персоналізованими порадами.

2. Нефункціональні вимоги. Продуктивність і надійність є важливими характеристиками. Бот повинен швидко реагувати на запити користувачів, забезпечувати високу точність відповіді та мати низький час затримки між введенням запиту та отриманням результату. Для цього необхідно використання хмарної інфраструктури, яка забезпечить достатню потужність для обробки запитів. Крім того, бот має бути доступний 24/7, що вимагає високої стабільності роботи серверів і належного моніторингу.

3. Інтерфейс користувача (UI). Інтерфейс бота має бути інтуїтивно зрозумілим, з можливістю легкої навігації та чіткими командами для користувачів. Важливо, щоб користувач міг швидко отримувати потрібні рекомендації без зайвих кроків. Для цього можуть бути реалізовані такі функції, як кнопки для швидкого вибору категорій запитів, а також можливість відправляти додаткові уточнення або задавати запитання.

4. Безпека і конфіденційність. Оскільки бот може отримувати і обробляти персональні дані користувачів, важливо забезпечити високу безпеку їх зберігання і передачі. Всі дані, які використовуються для персоналізації

рекомендацій, мають бути зашифровані, а доступ до них повинен бути обмежений відповідними заходами безпеки. Крім того, важливо реалізувати відповідність з правилами конфіденційності і захисту персональних даних.

5. Масштабованість і адаптивність. Телеграм-бот повинен бути здатний масштабуватися з ростом кількості користувачів, забезпечуючи стабільну роботу навіть при великих навантаженнях. Для цього варто використовувати технології, які дозволяють адаптувати систему до зростаючого обсягу запитів, такі як хмарні платформи або контейнеризація.

6. Інтеграція з іншими системами. Важливою вимогою є інтеграція бота з іншими сервісами та API, що забезпечать доступ до зовнішніх джерел даних, наприклад, для отримання інформації про погоду, медичні рекомендації або новини. Для цього необхідно налаштувати API з різними джерелами даних, щоб бот міг своєчасно та коректно реагувати на запити користувачів.

Усі ці вимоги разом забезпечать ефективну роботу та користувацьку зручність телеграм-бота “AI Advicer” для надання персоналізованих рекомендацій.

3.2Архітектура рішення

Розглянемо загальну структуру системи.

Архітектура рішення для телеграм-бота "AI Advicer" побудована таким чином, щоб забезпечити високу швидкість обробки запитів, персоналізацію рекомендацій та інтеграцію з різними зовнішніми API та технологіями ШІ. Система складається з кількох основних компонентів, включаючи Telegram API для взаємодії з користувачами, модуль ШІ для генерації рекомендацій, систему обробки тексту для інтерпретації запитів користувачів та обробку даних, а також хмарну інфраструктуру для забезпечення масштабованості та високої доступності.

Основні етапи роботи системи передбачають отримання запиту користувача через Telegram API, його обробку системою NLP для виділення

ключових елементів, передавання цих даних до модуля ШІ для генерації персоналізованих рекомендацій, після чого рекомендація повертається користувачу через Telegram-бота.

Опишемо компоненти системи.

Telegram API забезпечує основний канал комунікації між користувачем і ботом. Це дозволяє боту отримувати повідомлення від користувачів, обробляти їх та відправляти відповіді. Бот може використовувати різноманітні функції Telegram, такі як кнопки, меню, додавання медіа-файлів та інші інтерактивні елементи для покращення взаємодії з користувачами.

Модуль ШІ є основним елементом системи, який відповідає за надання персоналізованих рекомендацій користувачам. Модуль ШІ може бути реалізований на основі глибоких нейронних мереж або інших моделей машинного навчання, таких як колаборативне фільтрування або автокодери. Модуль ШІ аналізує дані, що надаються користувачем (наприклад, вік, інтереси, здоров'я, погода), і на основі цих даних генерує відповідні рекомендації. Цей модуль може також інтегрувати зовнішні дані, наприклад, прогнози погоди через API.

Виберемо технології та інструменти.

Telegram API є стандартним вибором для створення чат-ботів і забезпечує зручний доступ до функцій Telegram, таких як надсилання повідомлень, робота з кнопками, інтерактивними елементами та медіафайлами. Для роботи з API використовується популярна бібліотека python-telegram-bot, яка значно полегшує інтеграцію бота з Telegram.

Для реалізації модулю рекомендацій використовується метод машинного навчання **Random Forest** (Random Forest Classifier), який є **ансамблевим методом**. Він базується на побудові кількох дерев рішень (decision trees), які навчаються на різних підмножинах даних. Потім результати кожного дерева комбінуються для отримання фінального прогнозу.

Основні характеристики Random Forest: комбінує кілька моделей (дерев рішень), щоб покращити точність і зменшити ризик перенавчання; може

використовуватися як для задач класифікації, так і для регресії; кожне дерево тренується на випадковій підмножині даних, що допомагає зменшити кореляцію між деревами та покращити загальну стабільність моделі.

Модель Random Forest використовується для класифікації симптомів та передбачення відповідних рекомендацій.

Таким чином, вибір відповідних технологій і інструментів забезпечує ефективне, масштабоване і зручне рішення для створення телеграм-бота “AI Advicer”, здатного надавати високоякісні персоналізовані рекомендації користувачам.

3.3Розробка телеграм-бота “AI Advicer”

Розробка телеграм-бота “AI Advicer” включає кілька основних етапів, що забезпечують його функціональність, інтеграцію з модулями ШІ та обробку запитів користувачів. Це включає проектування архітектури, розробку інтерфейсу, налаштування взаємодії з Telegram API, інтеграцію з технологіями машинного навчання та NLP, а також тестування і деплой на сервер.

Взаємодія користувача та системи у Telegram-боті:

1. Початок роботи: користувач запускає Telegram-бот, отримуючи привітальне повідомлення з інструкціями щодо використання. Наприклад, бот пояснює, що потрібно ввести свій симптом, щоб отримати рекомендацію.

2. Введення симптомів: користувач вводить текстовий опис свого симптому, наприклад, "головний біль" або "сухий кашель". Обробка запиту: введений текст проходить попередню обробку та перетворюється на числовий індекс за допомогою енкодера, який було натреновано разом із моделлю; числовий індекс передається в натреновану модель Random Forest, яка здійснює класифікацію та прогнозує індекс відповідної рекомендації.

3. Формування відповіді: отриманий від моделі індекс рекомендації перетворюється назад у текстовий формат за допомогою зворотного енкодера;

Telegram-бот формує повідомлення з порадою, відповідною до введеного симптому.

4. Надання відповіді: бот надсилає користувачу повідомлення з рекомендацією. Наприклад: симптом: "головний біль"; рекомендація: "Випийте склянку води, відпочиньте в темному приміщенні. Якщо біль не минає, прийміть знеболювальне.".

5. Додатковий функціонал: якщо користувач вводить кілька симптомів або неточний опис, бот може попросити уточнити запит. У разі, якщо бот не може знайти рекомендацію, він пропонує звернутися до лікаря. Реалізовано додаткову команду /help для отримання довідки.

6. Завершення взаємодії: користувач може у будь-який момент завершити спілкування, використовуючи команду /stop, або просто закрити чат із ботом.

Ця система забезпечує зручну та швидку взаємодію, автоматизуючи процес надання порад за допомогою інтеграції машинного навчання та інтуїтивно зрозумілого інтерфейсу Telegram.

Для створення бота в Telegram та отримання доступу до API необхідно виконати кілька простих кроків.

Використаємо створення бота через BotFather.

BotFather – це офіційний бот Telegram, який дозволяє створювати та управляти іншими ботами в Telegram. Щоб створити нового бота, необхідно виконати такі кроки:

- відкрити Telegram і знайти бота BotFather за допомогою пошуку;
- натиснути "Start" для початку взаємодії;
- використовувати команду /newbot для створення нового бота;
- BotFather запитає ім’я для вашого бота. Ім’я має бути унікальним і не повторювати імена існуючих ботів;
- далі потрібно вказати юзернейм для бота. Він має бути унікальним, а також закінчуватися на "bot".

Ці кроки біло виконано. Після цього BotFather створив бота і надав токен API.

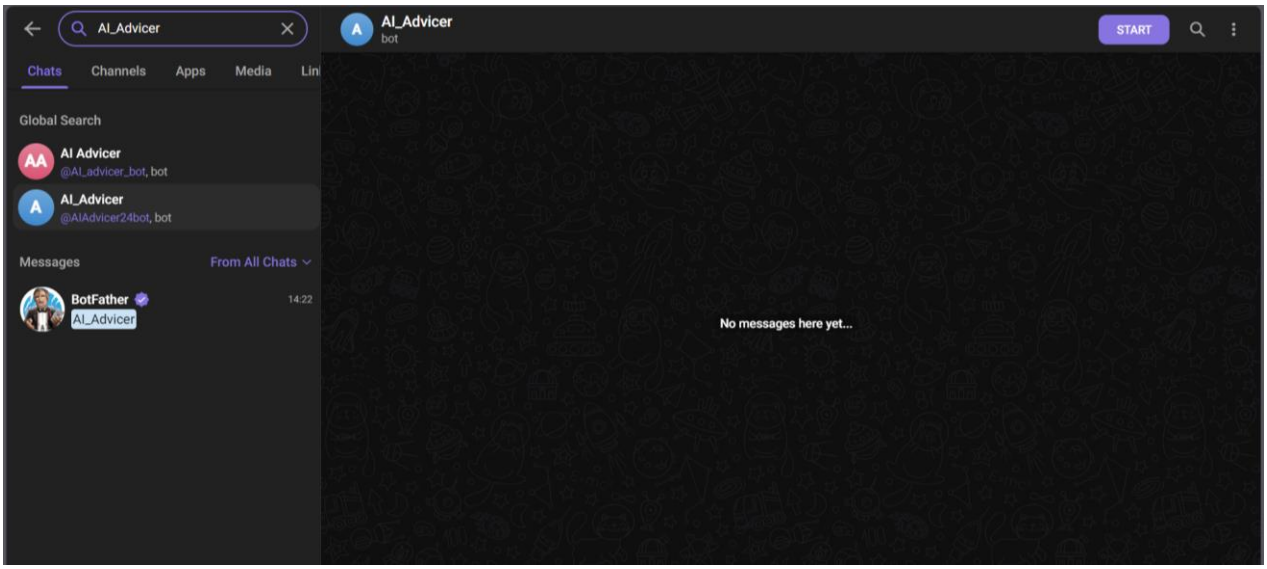

Рисунок 3.1 – Створений телеграм-бот

Після того як бот створено, BotFather надав токен, який є унікальним ідентифікатором для бота. Цей токен потрібно використовувати для доступу до Telegram API та взаємодії з вашим ботом. Важливо зберігати цей токен в безпеці, оскільки він дозволяє управляти ботом.

Тепер, маючи токен, можна почати інтеграцію бота з Telegram API. Для цього використаємо бібліотеки python-telegram-bot, telegram.

Важливо обробляти отриманий токен API обережно. Не публікувати його у відкритих репозиторіях або у місцях, де його можуть побачити інші користувачі. Зберігайти токен у конфігураційних файлах або середовищах, що забезпечують його безпечне використання.

Для частини програми, яка надає рекомендації відповідно до погодних умов, Збір даних про погоду на сервері OpenWeatherMap

OpenWeatherMap – це популярний сервіс для збору та обробки метеорологічних даних. Він надає API для доступу до інформації про поточну погоду, прогнозів, кліматичних умов та інших параметрів для будь-якої точки світу. Процес отримання даних із цього сервісу можна поділити на кілька кроків.

Потрібно почати з реєстрації та отримання API-ключа. API-ключ – це унікальний ідентифікатор, який використовується для авторизації запитів до сервера. Щоб отримати доступ до даних OpenWeatherMap, потрібно:

1. Зареєструватися на офіційному сайті сервісу OpenWeatherMap.
2. Увійти до особистого кабінету та створити новий API-ключ у розділі "API keys".

OpenWeatherMap надає кілька основних ендпоінтів для отримання даних:

1. Поточна погода:
   - URL: https://api.openweathermap.org/data/2.5/weather;
   - параметри:
     - q – назва міста (наприклад, q=London);
     - appid – ваш API-ключ;
     - units – одиниці вимірювання (наприклад, metric для градусів Цельсія).
2. Прогноз погоди на 5 днів:
   - URL: https://api.openweathermap.org/data/2.5/forecast;
   - використовується для отримання прогнозів з інтервалом у 3 години.
3. Кліматичні умови та історичні дані:
   - URL: https://api.openweathermap.org/data/2.5/onecall;
   - підтримує комплексний запит для отримання погодних даних, включаючи поточну погоду, денний та годинний прогнози, а також історичні дані.

Дані повертаються у форматі JSON (рис. 3.2).

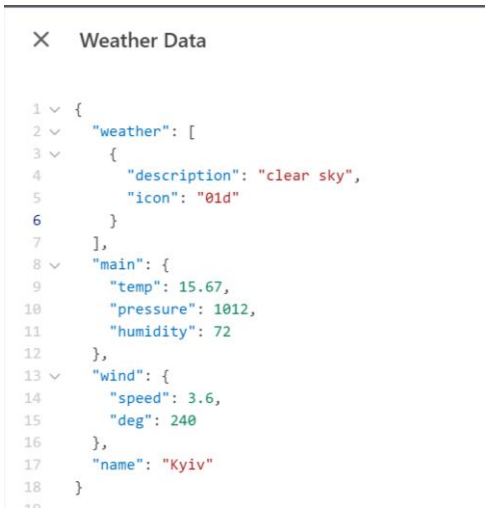

Рисунок 3.2 – Формат даних про погоду

Ключові параметри: weather.description – текстовий опис погодних умов; main.temp – температура; main.pressure – атмосферний тиск; main.humidity – вологість; wind.speed – швидкість вітру.

Зібрані дані можна інтегруємо в Telegram-бота для відображення прогнозів користувачам:

1. Надіслати запит до OpenWeatherMap API.
2. Отримати відповідь у форматі JSON.
3. Обробити дані та відформатувати повідомлення для користувача.
4. Відправити повідомлення через Telegram API.

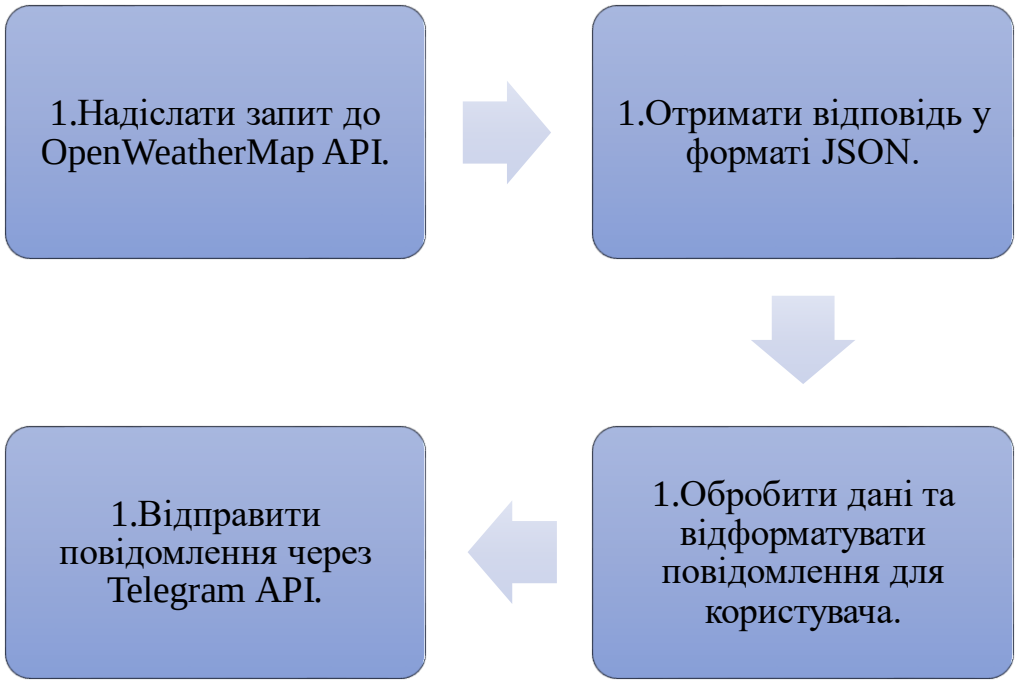

Рисунок 3.3 – Кроки взаємодії з OpenWeather

Логіка бота працює таким чином: бот приймає місцезнаходження чи запит і повертає інформацію про погоду та рекомендації щодо одягу.

Слід зауважити, що OpenWeatherMap має кілька тарифних планів: безкоштовний план обмежений кількістю запитів (60 запитів за хвилину); платні плани дозволяють отримувати більше даних із меншою затримкою. Ця платформа є універсальним інструментом для збору метеорологічних даних та їх інтеграції в різні проекти, включаючи Telegram-ботів.

Для створення моделі рекомендацій, яка надає поради відповідно до симптомів, важливо мати якісні та структуровані дані. Якщо реальні дані недоступні через конфіденційність або інші обмеження, можна використовувати симулятивні дані.

Симулятивні дані для навчання моделі рекомендацій були створені автоматично з метою моделювання взаємозв'язку між симптомами, можливими діагнозами та рекомендаціями. Генерація даних здійснювалася на основі типових сценаріїв, характерних для медичної діагностики та рекомендацій. Кожен запис у наборі даних включає:

– список симптомів, які описують стан користувача.

– рекомендації щодо дій, які необхідно виконати для поліпшення стану.

Дані були сформовані для широкого спектра ситуацій, таких як респіраторні захворювання, алергічні реакції, травми та загальні порушення здоров'я.

У програмі використовується метод машинного навчання Random Forest (Random Forest Classifier), який є ансамблевим методом. Він базується на побудові кількох дерев рішень (decision trees), які навчаються на різних підмножинах даних. Потім результати кожного дерева комбінуються для отримання фінального прогнозу.

Основні характеристики Random Forest:

– ансамблевий метод: комбінує кілька моделей (дерев рішень), щоб покращити точність і зменшити ризик перенавчання;

– класифікація та регресія: може використовуватися як для задач класифікації, так і для регресії;

– вибір випадкових підмножин: кожне дерево тренується на випадковій підмножині даних, що допомагає зменшити кореляцію між деревами та покращити загальну стабільність моделі.

Модель Random Forest використовується для класифікації симптомів та передбачення відповідних рекомендацій.

Опишемо побудову моделі. У загальному цей процес складається з таких кроків (рисунок 3.4):

1. Розділити дані на тренувальні та тестові набори.
2. Вибрати модель для навчання.
3. Навчити модель на тренувальних даних.
4. Оцінити модель на тестових даних.
5. Зберегти модель для подальшого використання.
6. Інтегрувати модель в Telegram-бота.
7. Протестувати бота та модель.

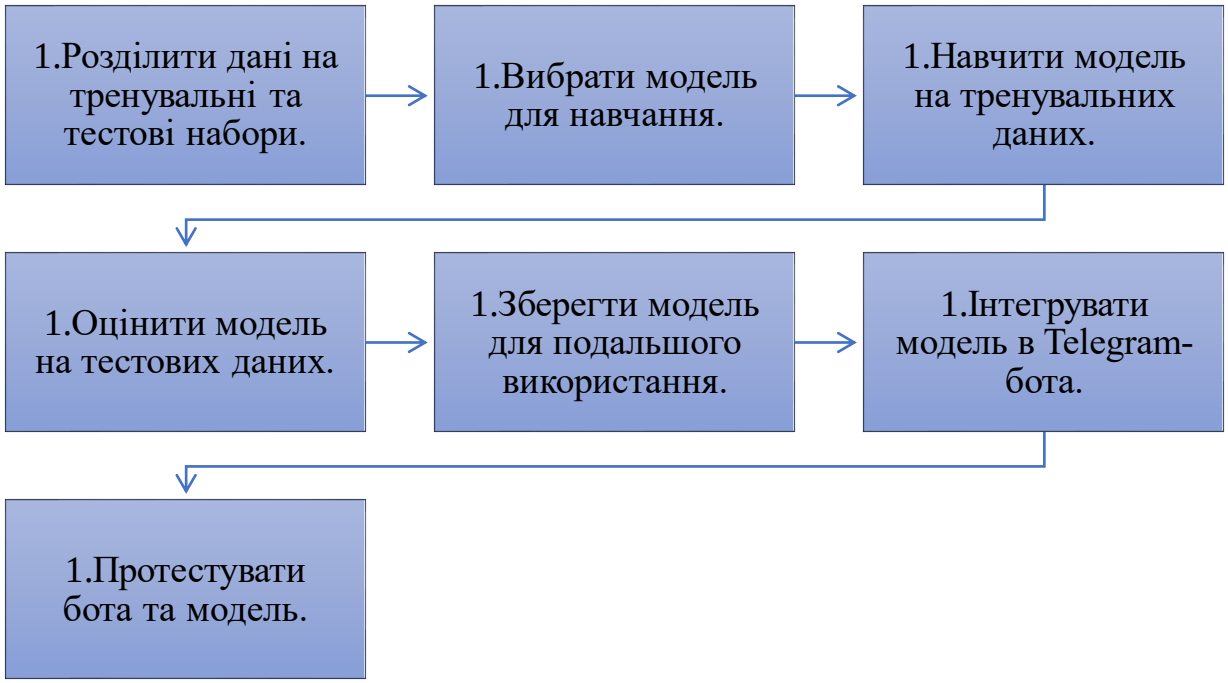

Рисунок 3.4 – Кроки реалізації модуля рекомендацій щодо стану здоров’я

Першим кроком є зчитування сирих даних з файлу raw_data.csv (рис. 3.5) і обробка тексту для створення індексів симптомів та рекомендацій. Для читання даних з raw_data.csv ми використовуємо pandas для читання сирих даних у форматі CSV. Для кожного стовпця тексту (симптоми і рекомендації) застосовуємо LabelEncoder, який перетворює категорії в числові індекси.

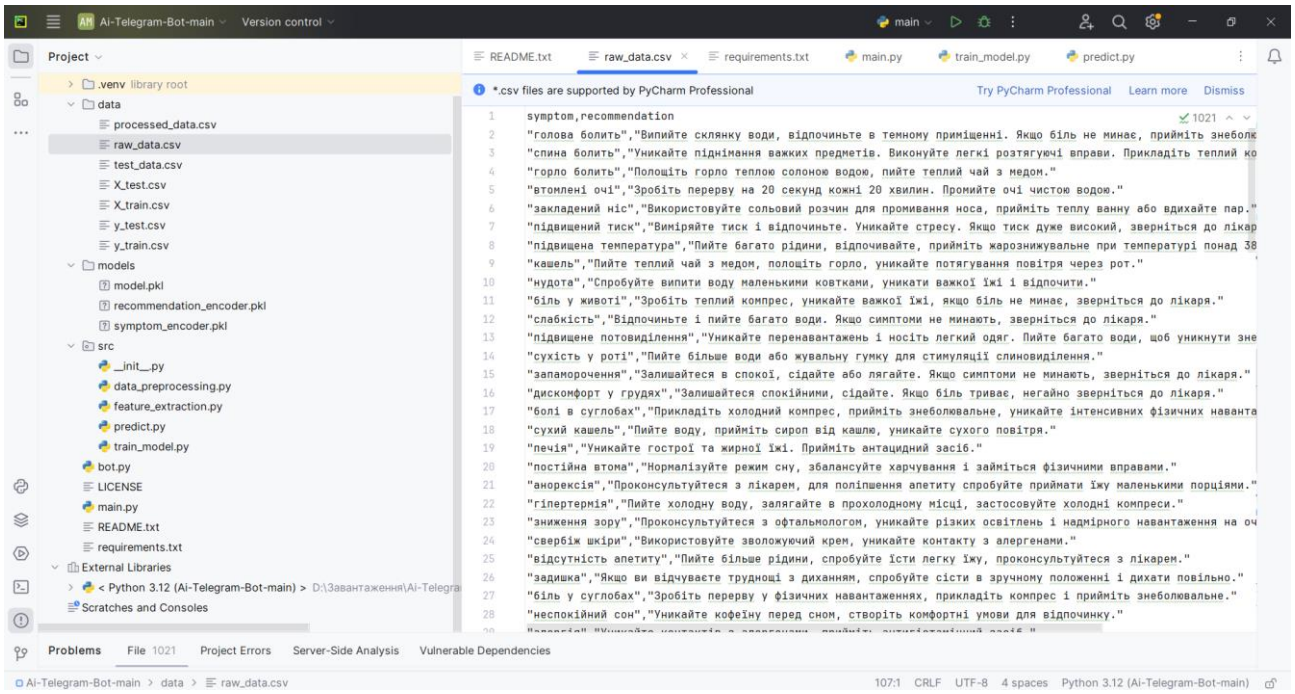

Рисунок 3.5 – Фрагмент файлу «raw_data.csv»

Для перетворення симптомів і рекомендацій використовуємо метод .transform() для перетворення текстових значень у числові індекси. Для створення нових стовпців у датафреймі після перетворення додаємо нові стовпці symptom_index та recommendation_index в наш датафрейм.

Створюємо новий CSV файл processed_data.csv (рис. 3.6), де зберігаються тільки індекси симптомів та рекомендацій.

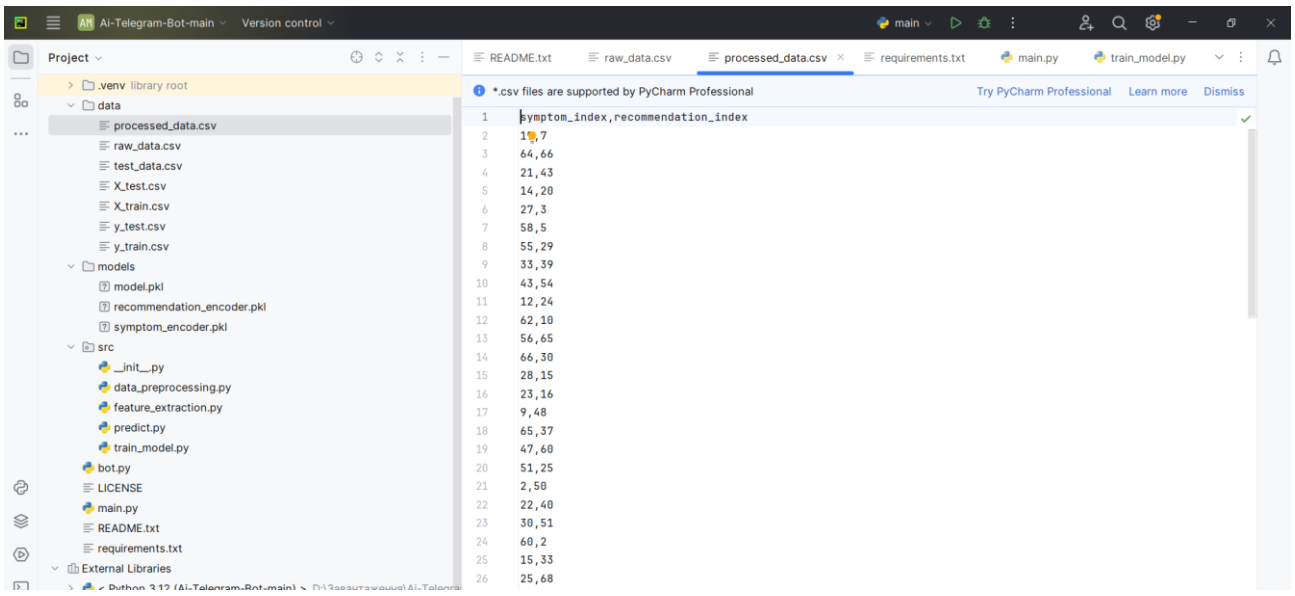

Рисунок 3.6 – Фрагмент файлу «processed_data.csv»

Після створення файлів processed_data.csv та test_data.csv (рис. 3.7), наступним кроком є підготовка даних для навчання моделі. Якщо дані ще не були поділені на тренувальний і тестовий набори, це необхідно зробити. Тренувальний набір використовується для навчання моделі, тоді як тестовий призначений для оцінки її ефективності.

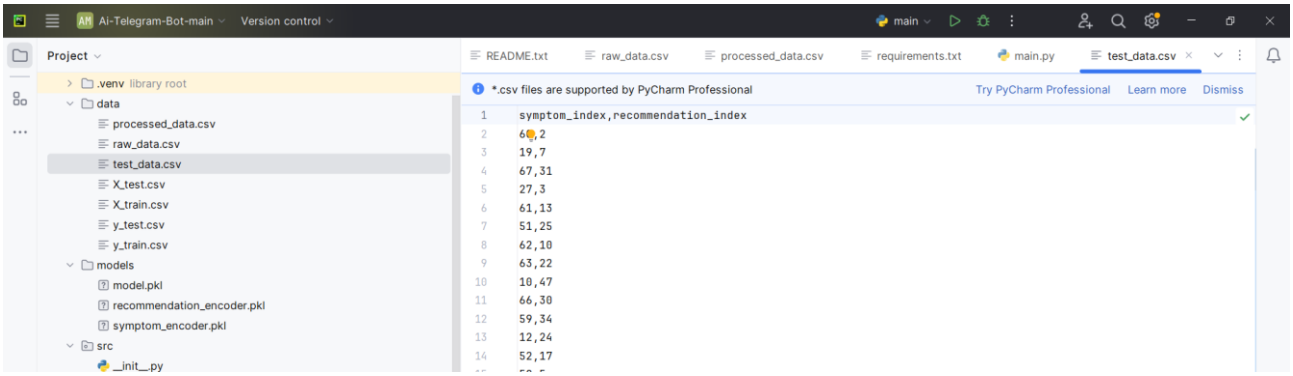

Рисунок 3.7 – Фрагмент файлу «test_data.csv»

У разі використання processed_data.csv, дані можна розділити на два набори: один для тренування моделі, а інший для тестування. Такий поділ забезпечує можливість перевірки точності моделі на нових даних після етапу навчання.

Для навчання моделі спочатку використовуються підготовлені дані, що зберігаються у файлах, таких як processed_data.csv, що містить індекси симптомів і відповідних рекомендацій. Першим кроком є розбиття даних на тренувальний і тестовий набори за допомогою функції train_test_split з бібліотеки sklearn.model_selection. Тренувальний набір використовується для навчання моделі, тоді як тестовий набір перевіряє точність і ефективність моделі. Для задачі класифікації, де ми передбачаємо рекомендації на основі симптомів, обрано модель Random Forest, яка є потужним методом ансамблевого навчання, що добре підходить для таких завдань. Після навчання модель оцінюється на тестових даних для перевірки її ефективності.

Інтеграція навченої моделі у телеграм-бот відбувається через створення скрипту, що обробляє запити користувачів. Після навчання моделі, вона

зберігається у файл за допомогою бібліотеки pickle, що дозволяє зберігати модель для подальшого використання. У бота завантажується ця модель, а також енкодери для перетворення симптомів у числові індекси. Коли користувач вводить симптоми, бот передає їх у модель після попередньої обробки. Модель прогнозує відповідні рекомендації, які бот повертає користувачу.

Для інтеграції з телеграм-ботом використовуються бібліотеки python-telegram-bot, яка надає зручний інтерфейс для роботи з API Telegram. Коли користувач вводить симптом, бот перетворює його в числовий індекс за допомогою енкодера, передає в модель, а потім виводить отриману рекомендацію. Це забезпечує автоматичний процес надання порад користувачам на основі введених симптомів.

Загальна структура проекту після навчання моделі подана на рисунку 3.8.

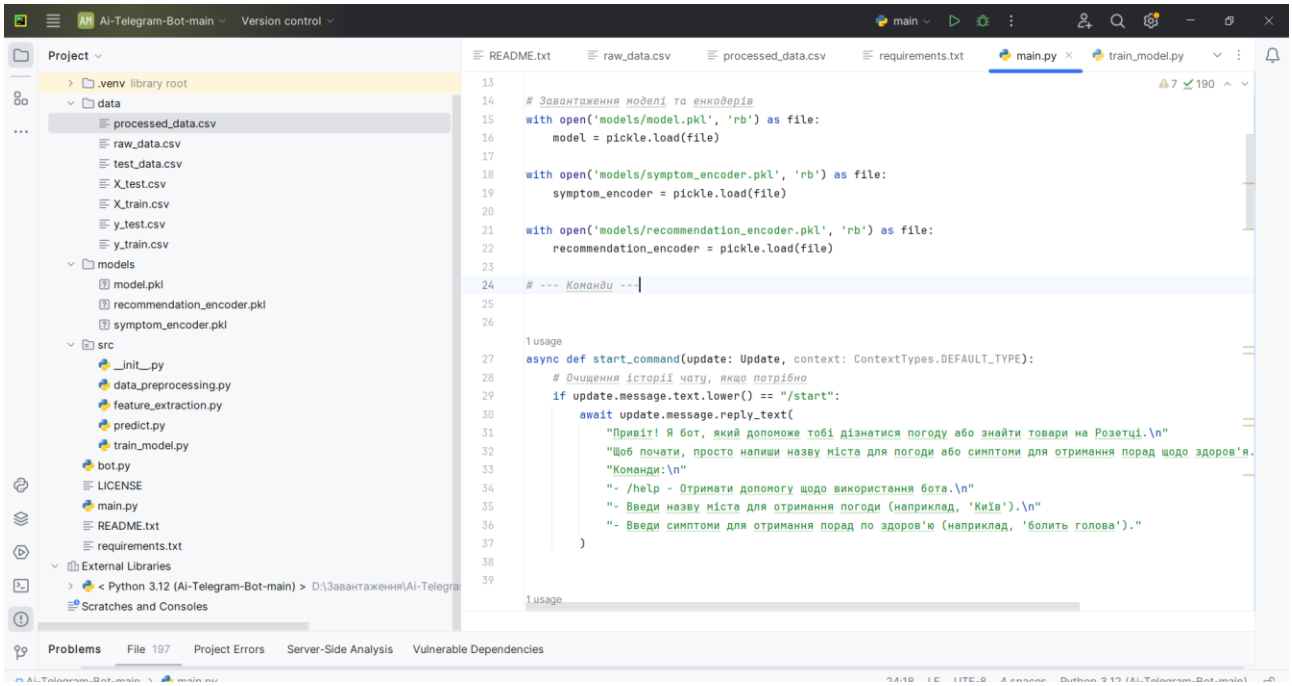

Рисунок 3.8 – Загальна структура проекту

Після тренування моделі, інтегруємо її у телеграм-бот. Програма складається з кількох основних функцій, що забезпечують інтерактивну взаємодію з користувачем. Перш за все, завантажуються необхідні моделі та енкодери, що використовуються для прогнозування рекомендацій на основі

симптомів користувача. Це включає моделі, які були збережені в файли за допомогою бібліотеки pickle. Бот обробляє команди /start і /help, надаючи користувачу інформацію про те, як користуватися ботом, а також можливість дізнатися погоду або отримати поради щодо здоров'я на основі симптомів.

Для отримання погоди бот використовує API OpenWeather, виконуючи запит до сервісу з переданим містом і отримує погодні умови, температуру, та рекомендації щодо одягу в залежності від температури. Користувач може ввести місто, і бот надасть актуальну інформацію.

Щодо здоров'я, бот аналізує симптоми, введені користувачем. Симптоми перетворюються в індекси за допомогою енкодера, і модель передбачає відповідні рекомендації. Результат відображається користувачу, допомагаючи йому отримати корисні поради для полегшення стану. Програма також реалізує обробку повідомлень від користувачів, щоб відповідати на запити щодо погоди або здоров'я. Всі ці функції забезпечують ефективну та зручну взаємодію між ботом і його користувачами.

3.4Тестування телеграм-бота “AI Advicer”

Тестування бота розпочалося з перевірки основних функцій, таких як отримання погоди та надання рекомендацій на основі симптомів. Першим етапом було тестування команд /start та /help (рис. 3.9), щоб переконатися, що бот правильно надає користувачу інформацію про доступні функції. Після успішного тестування команд бот виводив коректні повідомлення з описом доступних запитів і можливостей.

Далі було проведено тестування функції отримання погоди. Для цього тестувалося введення різних міст і перевірка точності отриманих даних про погодні умови. Бот успішно запитував API погоди, отримував актуальну інформацію та надавав рекомендації в залежності від температури.

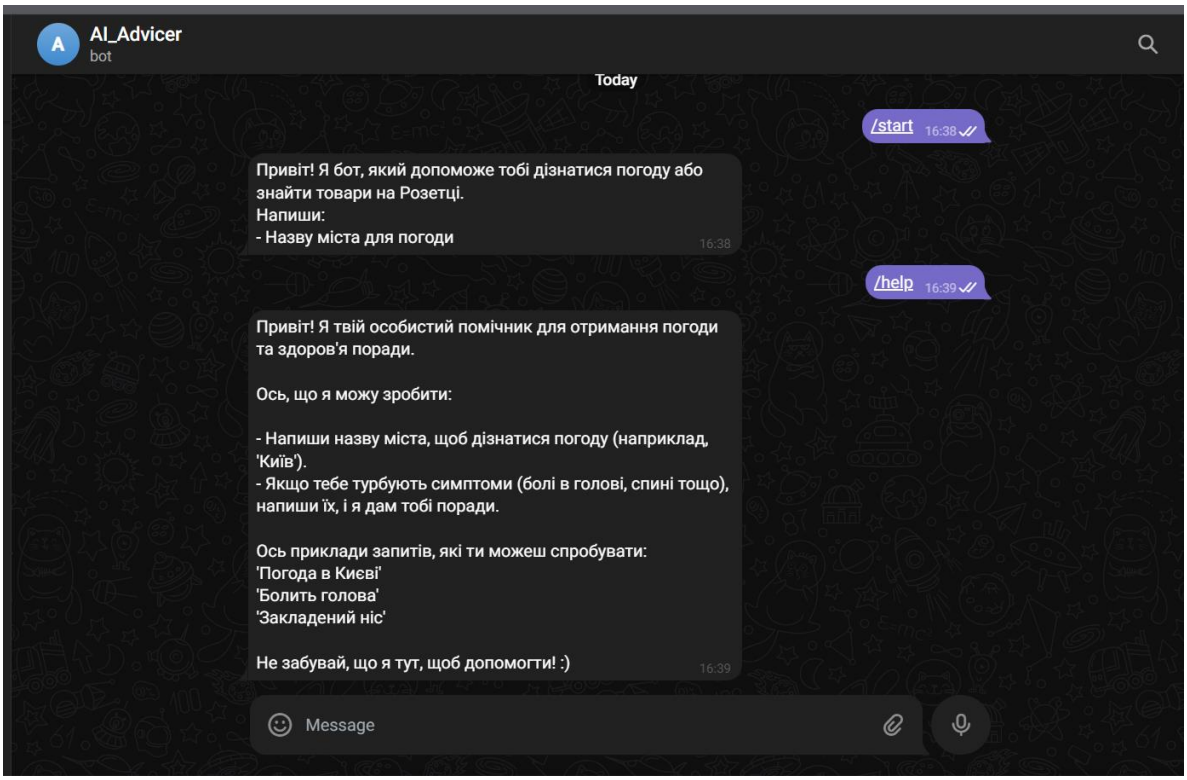

Рисунок 3.9 – Тестування команди /start та /help

Було перевірено також обробку помилок на випадок неправильних запитів або відсутності даних для певного міста, і бот коректно реагував на такі ситуації (рис. 3.10).

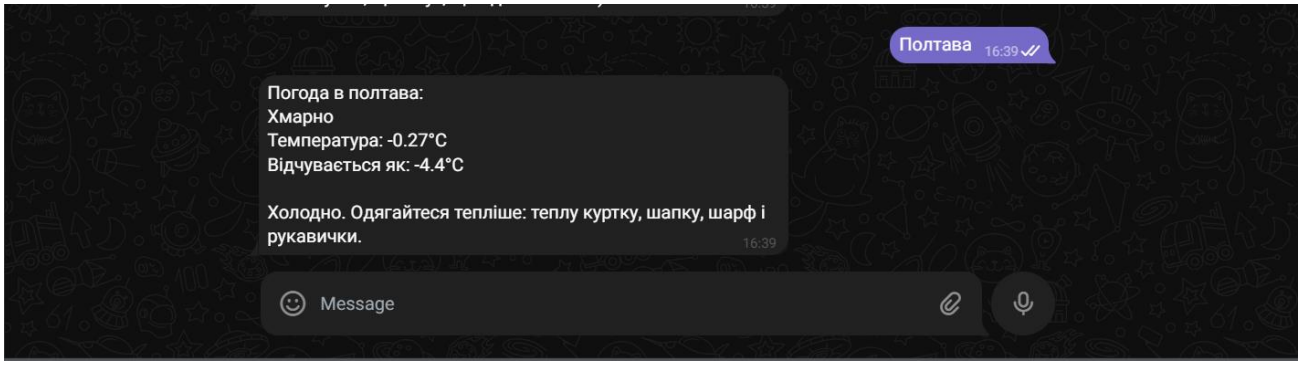

Рисунок 3.10 – Тестування отримання інформації про погоду

Тестування системи рекомендацій для здоров'я включало введення різних симптомів, таких як "болить голова" або "закладений ніс". Бот успішно перетворював введені симптоми на індекси, передавав їх моделі для прогнозу та надавав користувачам відповідні рекомендації. Всі отримані результати

відповідали очікуваному формату, і бот правильно виводив поради з медичної точки зору.

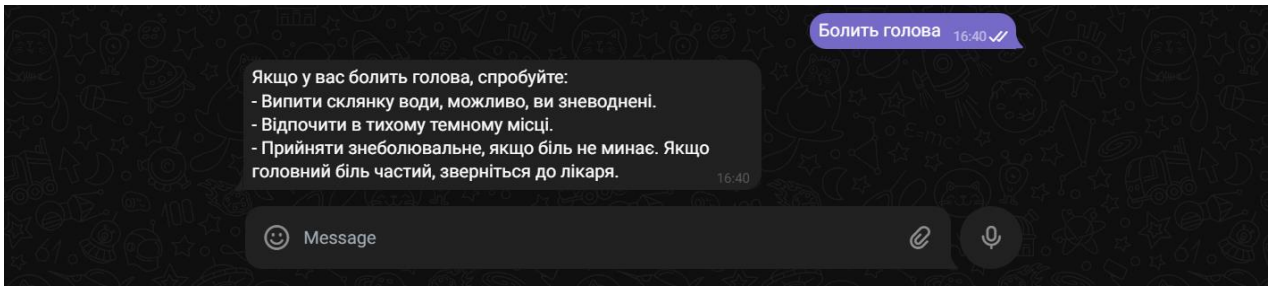

Рисунок 3.11 – Тестування отримання рекомендацій про здоров’я

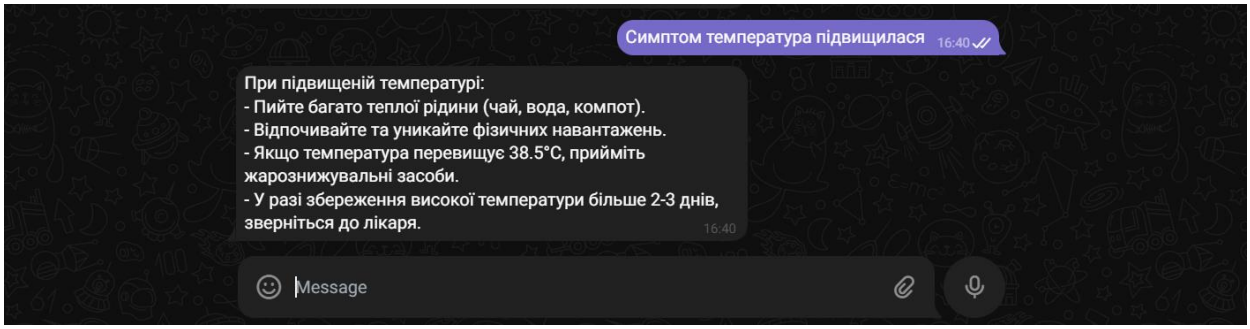

Рисунок 3.12 – Тестування отримання рекомендацій про здоров’я

Останнім етапом тестування було перевірка взаємодії бота з користувачами в реальному часі, включаючи перевірку його реакцій на різні запити та коректне відображення результатів. Також було здійснено тестування на різних пристроях і у різних мережах, щоб переконатися, що бот працює без збоїв. Тестування показало, що всі основні функції працюють стабільно, а бот надає точні та корисні відповіді.

ВИСНОВКИ

Розробка персонального помічника як елемента системи підтримки прийняття рішень є важливим кроком у напрямку автоматизації та оптимізації процесів у сучасному інформаційному середовищі. Використання Telegram-бота дозволяє інтегрувати систему в популярну платформу, що забезпечує зручний доступ і взаємодію з користувачем. Враховуючи здатність Telegram-ботів до інтеграції з іншими сервісами, така система може забезпечити комплексну підтримку рішень в різних сферах діяльності.

Класифікація систем підтримки прийняття рішень, зокрема за рівнем автоматизації, типом підтримуваних завдань та способом реалізації, дозволяє визначити основні функціональні можливості системи. Наприклад, інтерактивні СППР із високим рівнем взаємодії з користувачем забезпечують більш точні й контекстуалізовані рекомендації, що є важливим для розвитку персональних помічників. У той час як хмарні СППР дають змогу обробляти великі обсяги даних, що робить їх ідеальними для таких систем.

У результаті проведеної роботи було розроблено телеграм-бота, який поєднує в собі дві основні функції: надання погодної інформації та рекомендацій щодо здоров'я на основі симптомів, що були введені користувачем. Бот інтегрує машинне навчання для класифікації симптомів і надає корисні поради за допомогою розробленої моделі. Процес інтеграції був здійснений шляхом використання бібліотек Python, що забезпечили ефективну реалізацію необхідних алгоритмів для обробки даних, побудови моделі та взаємодії з користувачем через Telegram API.

Аналіз даних, що застосовувався під час розробки, дозволив обрати оптимальні технології та підходи для обробки текстових даних та побудови моделей машинного навчання. Завдяки використанню алгоритмів, таких як Random Forest, вдалося досягти високої точності в прогнозуванні рекомендацій на основі симптомів. Модель показала хороші результати в реальних умовах

тестування, що підтвердило її здатність до надання корисних і точних порад користувачам.

Тестування бота виявило високий рівень його ефективності та надійності в різних сценаріях використання. Користувачі могли легко отримати інформацію про погоду в будь-якому місті, а також отримати медичні рекомендації на основі введених симптомів. Всі функціональні можливості бота працювали без збоїв, а також було перевірено, що він коректно обробляє помилки, що виникають у випадку неправильних запитів або відсутності даних.

На основі виконаного дослідження можна зробити висновок, що розроблений бот є корисним інструментом для автоматизації надання рекомендацій та збору інформації через інтерактивний інтерфейс. У майбутньому можливе вдосконалення цієї системи шляхом додавання нових функцій, таких як підтримка більшої кількості симптомів і рекомендацій або інтеграція з іншими API для розширення можливостей бота.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Барановський, Дмитро. (2023). Розробка системи підтримки прийняття рішень сімейного лікаря на засадах системного аналізу. Технічна інженерія. 89-95. 10.26642/ten-2023-2(92)-89-95.

2. Бределєв І. В., Ковалюк О. О. Розробка персональної системи підтримки прийняття рішень на основі мобільних технологій. Матеріали LIII науково-технічної конференції підрозділів ВНТУ, Вінниця, 20-22 березня 2024 р. Електрон. текст. дані. 2024. URI: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20930.

3. Розробка ефективних алгоритмів аналізу даних. URL: https://ela.kpi.ua/server/api/core/bitstreams/aec59977-a426-4b61-9895-26f6d626b24b/content (дата звернення: 20.09.2024)

4. Оптимізація продуктивності системи на мобільних пристроях. URL: https://outsourcing.team/uk/blog/smm-blog/yak-optimizuvati-sajt-dlya-mobilnih-pristroyiv/ (дата звернення: 20.09.2024).

5. Мобільні технології та їх переваги. URL: https://smile-ukraine.com/ua/mobile-apps/mobile- technology/introduction (датазвернення: 20.03.2024).

6. Borysova, H.V. and Yakovenko, A.V. (2016), «Systemy pidtrymky pryiniattia rishen v medychnii diahnostytsi, Aktualni zadachi suchasnykh tekhnolohii, No. 5, pp. 16–17.

7. Bilak, N.V. (2023), Osnovy systemnoho analizu ta pryiniattia rishennia, NAU, Kyiv, 81 p.

8. Radzishevska, Ye.B. and Vysotska. O.V. (2019), Informatsiini tekhnolohii v medytsyni E-Health, KhNMU, Kharkiv, 73 p.

9. Sutton, R.T., Pincock, D., Baumgart, D.C. et al. (2022), «An overview of clinical decision support systems: benefits, risks, and strategies for success», npj

Digital Medicine, [Online], available at: https://www.nature.com/articles/s41746-020-0221-y

10. Коляструк, Богдан, Антонов, Юрій. (2020). Особливості розробки Telegram бота для інформаційної підтримки навчального процесу. 10.36074/02.10.2020.v1.15.

11. Що таке багатошаровий персептрон (Multilayer Perceptron, MLP) у машинному навчанні? URL: https://cutt.ly/Ce3eD2GA. (дата звернення – 18.11.24).

12. Різник О. М. Динамічні рекурентні нейронні мережі // ММС. 2009. №3. URL: https://cyberleninka.ru/article/n/dinamichni-rekurentni-neyronni-merezhi (дата звернення – 20.11.24).

13. Калбазов Д.Й., Данченко О.І., Лисецький Ю.М. НЕЙРОМЕРЕЖІ. РОЗВИТОК ТА ПЕРСПЕКТИВИ. Математичні машини і системи. 2024. No 2. С. 24–32.

14. Recurrent Neural Network. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/recurrent-neural-network. (дата звернення – 18.11.24).

15. Проценко Антон. Як працює нейронна мережа і її застосування. URL: https://proit.com.ua/news/yak-pratsyuye-nejronna-merezha-yiyi/ . (дата звернення – 17.11.24).

16. Collaborative Filtering in Machine Learning. URL: https://www.geeksforgeeks.org/collaborative-filtering-ml/ (дата звернення – 20.11.24).

17. Collaborative Filtering In Machine Learning Made Simple [6 Different Approaches]. URL: https://spotintelligence.com/2024/04/25/collaborative-filtering/ (дата звернення – 20.11.24).

18. Popular Python AI Libraries. URL: https://www.restack.io/p/ai-python-answer-popular-python-ai-libraries-cat-ai (дата звернення – 18.11.24).

19. 10 Best Python Libraries for Machine Learning & AI. URL: https://www.unite.ai/10-best-python-libraries-for-machine-learning-ai/.(дата звернення – 19.11.24).

20. python-telegram-bot v21.10. URL: https://docs.python-telegram-bot.org/en/v21.10/.(дата звернення – 18.11.24).

21. Telegram Bot API. URL: https://core.telegram.org/bots/api/ (дата звернення – 18.11.24).

22. Weather API. URL: https://openweathermap.org/api. (дата звернення – 18.11.24).

23. Машинне навчання: навчальний посібник призначений для студентів, що навчаються за першим (бакалаврським) рівнем вищої освіти за спеціальностями галузі знань 12 „Інформаційні технології”. 3-тє видання, стереотипне / за науковою редакцією д.т.н., проф., В.В. Пасічника. – Львів: Видавництво «Новий Світ – 2000», 2024. – 330 с.

24. Машинне навчання: комп’ютерний практикум з дисципліни «Машинне навчання» [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)/ Л.М. Олещенко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 92 с.