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 дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему
Розробка чат-бота для футбольних фанатів
з використанням Natural Language Processing (NLP)

Виконав: студент 6 курсу, групи 601-ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Скадченко Р. Р.
(прізвище та ініціали)
Керівник Двірна О. А.
(прізвище та ініціали)

Полтава – 2025 року

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 чат-бота для футбольних фанатів з використанням Natural Language Processing»

Студента групи 601-ТН Скадченка Ростислава Романовича

Керівник роботи
кандидат фізико-математичних наук,
доцент Двірна О. А.

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

Полтава – 2025 року

РЕФЕРАТ

Кваліфікаційна робота магістра: 69 с., 42 рис., 3 табл., 18 джерел, 2 додатки.

Об'єкт дослідження: система інтерактивної взаємодії з футбольними фанатами за допомогою чат-бота на основі обробки природної мови (NLP).

Мета роботи: розробка чат-бота, який забезпечує якісну обробку текстових запитів футбольних фанатів, надає інформацію про футбольні події та аналізує запити з використанням NLP.

Методи дослідження: обробка природної мови (токенізація, лематизація, видалення стоп-слів), техніки векторизації тексту (CountVectorizer, TF-IDF), алгоритми машинного навчання (Logistic Regression, Support Vector Machine, Random Forest), бібліотеки для створення чат-ботів (Dialogflow, Rasa, Streamlit).

Структура та обсяг: магістерська кваліфікаційна робота складається зі вступу, трьох розділів та висновків.

У першому розділі розглянуто теоретичні основи розробки чат-ботів, включаючи методи обробки природної мови, алгоритми машинного навчання та сучасні підходи до створення інтерактивних систем взаємодії.

Другий розділ присвячено постановці задачі, вибору методів обробки текстових даних, розробці структури чат-бота та об'рунтуванню використаних технологій.

У третьому розділі представлено реалізацію чат-бота для футбольних фанатів, включаючи навчання моделей, інтеграцію функціоналу з футбольними базами даних, тестування роботи чат-бота та аналіз результатів його взаємодії з користувачами.

Ключові слова: футбольний чат-бот, машинне навчання, обробка природної мови, NLP, TF-IDF, SVM, токенізація, лематизація, класифікація текстів, інтерактивна система, футбольні фанати, Rasa, Dialogflow.

ABSTRACT

Qualification work of a master: 69 pages, 42 figures, 3 tables, 18 references, 2 appendices.

Object of research: an interactive system for engaging football fans using a chatbot based on natural language processing (NLP).

Purpose of the work: to develop a chatbot capable of efficiently processing text queries from football fans, providing information about football events, and analyzing requests using NLP.

Research methods: natural language processing (tokenization, lemmatization, stop-word removal), text vectorization techniques (CountVectorizer, TF-IDF), machine learning algorithms (Logistic Regression, Support Vector Machine, Random Forest), libraries for chatbot development (Dialogflow, Rasa, Streamlit).

Structure and scope: the master's qualification thesis consists of an introduction, three chapters, and conclusions.

The first chapter reviews the theoretical foundations of chatbot development, including natural language processing methods, machine learning algorithms, and modern approaches to creating interactive engagement systems.

The second chapter is dedicated to the problem statement, selection of text data processing methods, development of the chatbot structure, and justification of the chosen technologies.

The third chapter presents the implementation of a chatbot for football fans, including model training, integration of functionality with football databases, testing of the chatbot's performance, and analysis of its interaction results with users.

Keywords: football chatbot, machine learning, natural language processing, NLP, TF-IDF, SVM, tokenization, lemmatization, text classification, interactive system, football fans, Rasa, Dialogflow.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 6

ВСТУП 7

РОЗДІЛ 1 ОСНОВИ РОЗРОБКИ ЧАТ-БОТА ДЛЯ ФУТБОЛЬНИХ ФАНАТІВ З ВИКОРИСТАННЯМ NATURAL LANGUAGE PROCESSING (NLP) 9

1.1 Визначення чат-ботів та їх значення у цифрових комунікаціях 9

1.2 Принципи обробки природної мови у взаємодії з користувачами 13

1.3 Алгоритми машинного навчання для створення інтерактивних систем 17

1.4 Особливості роботи з текстами футбольної тематики 19

1.5 Висновки до Розділу 1 22

РОЗДІЛ 2 МЕТОДИКА ТА ЗАСОБИ РОЗРОБКИ ЧАТ-БОТА ДЛЯ ФУТБОЛЬНИХ ФАНАТІВ 25

2.1 Постановка задачі та формулювання вимог до системи 25

2.2 Вибір методів обробки текстових даних 28

2.3 Архітектура системи чат-бота 31

2.4 Інтеграція з футбольними інформаційними ресурсами 34

2.5 Інструменти для розробки та тестування 37

2.6 Висновки до Розділу 2 41

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ 44

3.1 Етапи реалізації функціоналу чат-бота 44

3.2 Підготовка текстових даних до обробки 46

3.3 Навчання моделей та інтеграція алгоритмів NLP 50

3.4 Тестування чат-бота та оцінка його ефективності 54

3.5 Висновки до Розділу 3 57

ВИСНОВКИ 60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62
ДОДАТОК А ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ ПРОГАМИ ... 63
ДОДАТОК Б РЕЗУЛЬТАТИ РОБОТИ ІНТЕРАКТИВНОГО ДОДАТКУ 69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML – Machine Learning.

NLP – Natural Language Processing.

TF-IDF – Term Frequency-Inverse Document Frequency.

TF – Term Frequency.

IDF – Inverse Document Frequency.

SGD – Stochastic Gradient Descent.

SVM – Support Vector Machine.

BoW – Bag of Words.

BERT – Bidirectional Encoder Representations from Transformers.

SMOTE – Synthetic Minority Oversampling Technique.

LSTM – Long Short-Term Memory.

API – Application Programming Interface.

CSV – Comma-Separated Values.

ВСТУП

Актуальність. Розробка інтерактивних інструментів для взаємодії з користувачами за допомогою чат-ботів набуває дедалі більшого значення в сучасному інформаційному суспільстві. Зростання популярності футболу як одного з найулюбленіших видів спорту у світі зумовило потребу у створенні інноваційних рішень для залучення футбольних фанатів та підтримки їхньої активності. Інтерактивні системи, засновані на технологіях обробки природної мови (NLP), дозволяють забезпечити якісний і персоналізований досвід для користувачів, надаючи їм можливість отримувати актуальну інформацію про матчі, команди, результати та інші події у світі футболу.

Застосування алгоритмів машинного навчання та сучасних методів обробки тексту відкриває нові горизонти для автоматизації обробки запитів користувачів, враховуючи специфіку їхнього стилю спілкування, сленг і контекст. Технології NLP дозволяють створювати чат-ботів, здатних відповідати на запити з високою точністю, інтерактивно реагувати на потреби користувачів і адаптуватися до їхніх уподобань. Це особливо актуально для футбольної спільноти, де швидкий доступ до інформації та взаємодія відіграють ключову роль у підтримці інтересу фанатів.

Актуальність цієї роботи зумовлена також потребою у створенні інтегрованих рішень, які б дозволяли поєднати бази даних футбольної статистики з можливостями інтерактивної взаємодії. Чат-боти для футбольних фанатів можуть слугувати не лише джерелом інформації, але й платформою для аналізу даних, підтримки комунікації між фанатами та покращення їхнього досвіду взаємодії із сучасними цифровими технологіями.

Мета роботи. Метою роботи є розробка теоретичної та практичної основи для створення чат-бота, який забезпечує якісну взаємодію з футбольними фанатами, використовуючи технології обробки природної мови та машинного навчання. Особлива увага приділяється аналізу текстових запитів, їхній

класифікації та інтеграції функціональності, що дозволяє надавати релевантну інформацію в реальному часі.

Обґрунтування проектних рішень. Для досягнення поставленої мети використовуються сучасні методи NLP, включаючи токенізацію, лематизацію, видалення стоп-слів, а також техніки векторизації тексту (TF-IDF). Алгоритми машинного навчання, такі як Logistic Regression, SVM та Random Forest, забезпечують ефективну обробку текстових даних, що є ключовим для досягнення високої точності у відповідях чат-бота. Створення інтерактивного інтерфейсу для взаємодії з користувачами реалізується за допомогою бібліотек Streamlit та Rasa.

Галузі застосування. Результати роботи можуть бути застосовані у спортивній індустрії, маркетингу, розробці цифрових платформ для вболівальників, а також у дослідницьких цілях для аналізу поведінки футбольної аудиторії. Чат-боти можуть сприяти розвитку комунікації між клубами та їхніми фанатами, а також покращувати доступ до актуальної інформації для користувачів.

Таким чином, ця робота спрямована на підвищення ефективності взаємодії між футбольними фанатами та цифровими платформами, використовуючи сучасні технології обробки природної мови та машинного навчання. У першому розділі досліджено теоретичні аспекти створення чат-ботів, у наступних розділах представлено практичну реалізацію розробленого рішення, аналіз його роботи та перспективи впровадження.

РОЗДІЛ 1
ОСНОВИ РОЗРОБКИ ЧАТ-БОТА ДЛЯ ФУТБОЛЬНИХ ФАНАТІВ З ВИКОРИСТАННЯМ NATURAL LANGUAGE PROCESSING (NLP)

1.1 Визначення чат-ботів та їх значення у цифрових комунікаціях

Чат-боти (англ. chatbots) — це програмні системи, створені для автоматизованої взаємодії з користувачами через текстові або голосові інтерфейси. Вони базуються на технологіях обробки природної мови (NLP) та машинного навчання і використовуються для різноманітних завдань, від підтримки клієнтів до персоналізованих рекомендацій [1].

Основні характеристики чат-ботів

Чат-боти мають низку характеристик, що роблять їх незамінними в цифрових комунікаціях:

1. **Інтерактивність:** Здатність відповідати на запити користувачів у режимі реального часу. Це важливо для забезпечення швидкого доступу до інформації та підвищення рівня обслуговування.
2. **Гнучкість:** Чат-боти можуть адаптуватися до різних тематик, від загальних інформаційних запитів до спеціалізованих задач, таких як консультації з медичних питань чи технічна підтримка.
3. **Масштабованість:** Один чат-бот здатний одночасно обслуговувати тисячі користувачів, що значно знижує навантаження на людський персонал.
4. **Доступність:** Вони працюють 24/7, забезпечуючи безперервну взаємодію з користувачами.

Типи чат-ботів:

1. **Правиломодульні (rule-based):** Ці чат-боти діють за заздалегідь визначеними сценаріями. Вони прості у реалізації, але мають обмежену функціональність:
    - наприклад, чат-бот банку може надати інформацію про курс валют або графік роботи відділень.
2. **Інтелектуальні (AI-powered):** Використовують алгоритми машинного навчання для розуміння контексту запитів та формування відповідей. Вони здатні навчатися та адаптуватися до нових даних:
    - наприклад, віртуальний асистент Siri від Apple здатний відповідати на складні запити, враховуючи контекст попередніх взаємодій.

Значення чат-ботів у цифрових комунікаціях

Чат-боти відіграють важливу роль у сучасному цифровому середовищі завдяки їх здатності спрощувати процеси комунікації та автоматизувати рутинні завдання. Нижче наведено основні сфери їх застосування:

1. **Автоматизація обслуговування клієнтів.** Чат-боти дозволяють швидко вирішувати типові запити клієнтів, зменшуючи навантаження на служби підтримки. Наприклад, чат-бот авіакомпанії може допомогти з перевіркою статусу рейсу чи зміною бронювання, що зменшує час очікування для клієнтів.
2. **Зниження витрат.** Завдяки автоматизації багатьох процесів, компанії скорочують витрати на обслуговування клієнтів. Згідно з дослідженням компанії Gartner, до 2025 року використання чат-ботів може знизити витрати компаній на обслуговування клієнтів на 30%.
3. **Покращення користувацького досвіду.** Користувачі отримують можливість швидко отримувати відповіді на свої запити в будь-який час доби. Наприклад, в інтернет-магазинах чат-боти можуть допомогти знайти потрібний товар, порівняти ціни або оформити замовлення, що підвищує задоволеність клієнтів.

4. **Персоналізація.** Інтелектуальні чат-боти можуть використовувати дані про попередні запити користувачів для створення персоналізованих рекомендацій. Наприклад, Spotify пропонує музичні плейлисти, які відповідають вподобанням конкретного користувача.
5. **Підвищення залученості аудиторії.** У спортивній індустрії чат-боти активно використовуються для взаємодії з фанатами. Наприклад, офіційний чат-бот ФК "Барселона" надає актуальну інформацію про матчі, статистику гравців і дозволяє брати участь у голосуваннях.

Приклади успішного використання

1. **Duolingo:** Чат-бот для вивчення мов, який допомагає користувачам практикувати розмовну мову.
2. **Starbucks Barista Bot:** Дозволяє клієнтам замовляти каву через месенджери.
3. **Sephora Virtual Artist:** Бот у Facebook Messenger, який допомагає вибрати косметику та надає консультації з її використання.

Проблеми та виклики

Попри численні переваги, чат-боти мають певні обмеження:

- відсутність глибокого розуміння складного контексту;
- складність у роботі з двозначними або багатозначними запитами;
- залежність від якості початкових даних та сценаріїв.

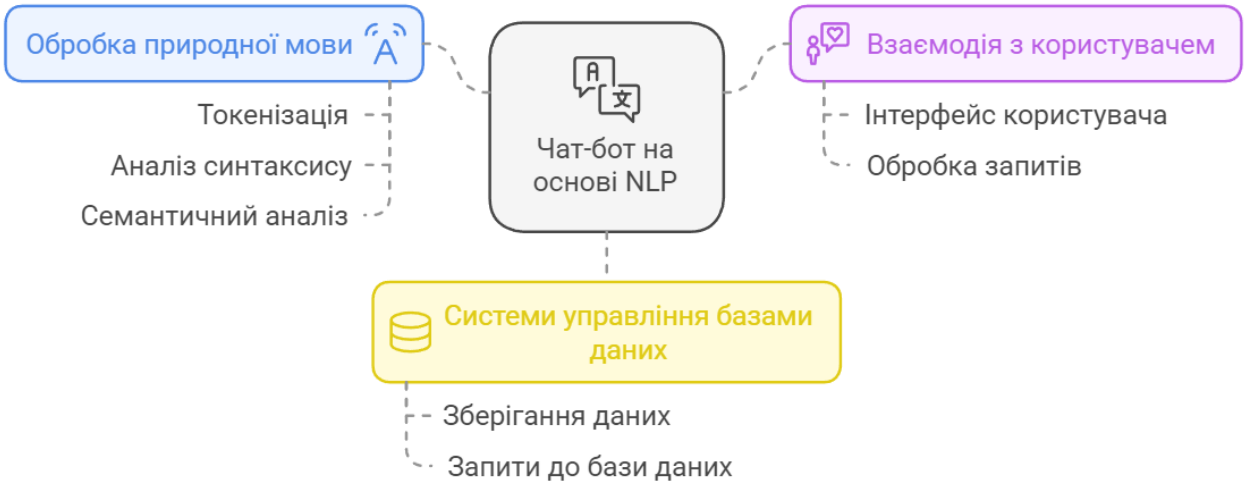

Рисунок 1.1 - Архітектура чат-бота на основі NLP

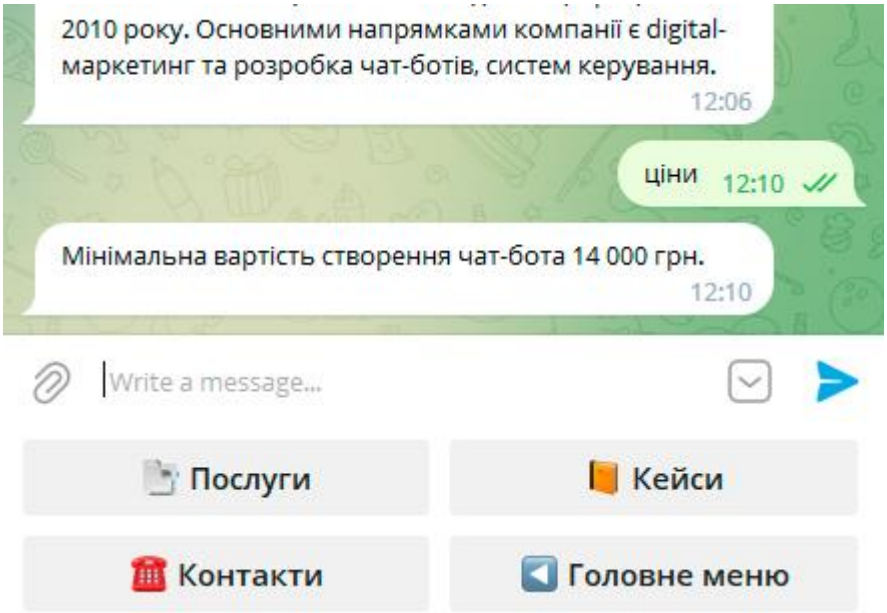

Рисунок 1.2 - Взаємодія чат-бота з користувачем у месенджері

Чат-боти є ключовим інструментом у цифрових комунікаціях, дозволяючи автоматизувати процеси, покращити якість обслуговування та підвищити лояльність клієнтів. У контексті футбольної індустрії чат-боти мають великий потенціал для створення інтерактивного досвіду, підтримки фанатів та аналізу

їхньої активності. Подальший розвиток технологій NLP та машинного навчання забезпечить ще більшу ефективність і функціональність цих систем.

1.2 Принципи обробки природної мови у взаємодії з користувачами

Обробка природної мови (Natural Language Processing, NLP) — це галузь штучного інтелекту, яка займається аналізом і синтезом людської мови для створення систем, здатних взаємодіяти з користувачами на їхній рідній мові. У контексті чат-ботів NLP є основою для обробки запитів, формування відповідей та адаптації до потреб користувача.

Основні етапи обробки природної мови:

1. **Збір текстових даних.** На цьому етапі система отримує вхідний текст у вигляді запиту користувача. Це може бути повідомлення в месенджері, текстовий запит або голосове повідомлення, яке перетворюється в текст.
2. **Попередня обробка тексту.** Попередня обробка дозволяє стандартизувати текст та підготувати його до подальшого аналізу:
    - **токенізація:** Розбиття тексту на окремі слова або фрази (токени);
    - **лематизація:** Приведення слів до їх базової форми (наприклад, "бігти" замість "бігав");
    - **видалення стоп-слів:** Усунення загальних слів ("і", "але", "як"), які не несуть значної інформації;
    - **очищення тексту:** Видалення спеціальних символів, цифр або зайвих пробілів.
3. **Векторизація тексту.** Для того, щоб текст міг бути оброблений алгоритмами машинного навчання, його потрібно перетворити у числове представлення. Основні методи:
    - **Bag of Words (BoW):** Текст перетворюється у вектор, що відображає частоту кожного слова;

- **TF-IDF (Term Frequency-Inverse Document Frequency):** Враховує важливість слова у контексті певного документа;
- **Word Embeddings:** Використання попередньо навчених моделей, таких як Word2Vec або GloVe, для отримання багатовимірних векторів, що зберігають семантичні зв’язки між словами.

4. **Аналіз тексту.** На цьому етапі алгоритми аналізують текст для визначення його змісту чи контексту. Наприклад:
    - **класифікація:** Визначення категорії запиту (наприклад, запитання, скарга, подяка);
    - **витяг сутностей:** Виділення ключових елементів тексту (імена, дати, локації);
    - **аналіз настрою:** Визначення емоційного забарвлення тексту (позитивне, негативне, нейтральне).
5. **Формування відповіді.** На основі аналізу система генерує відповідь. Це може бути:
    - вибір готової відповіді з бази знань;
    - генерація тексту за допомогою мовних моделей (наприклад, GPT);
    - комбінація обох методів для забезпечення точності та релевантності [2].

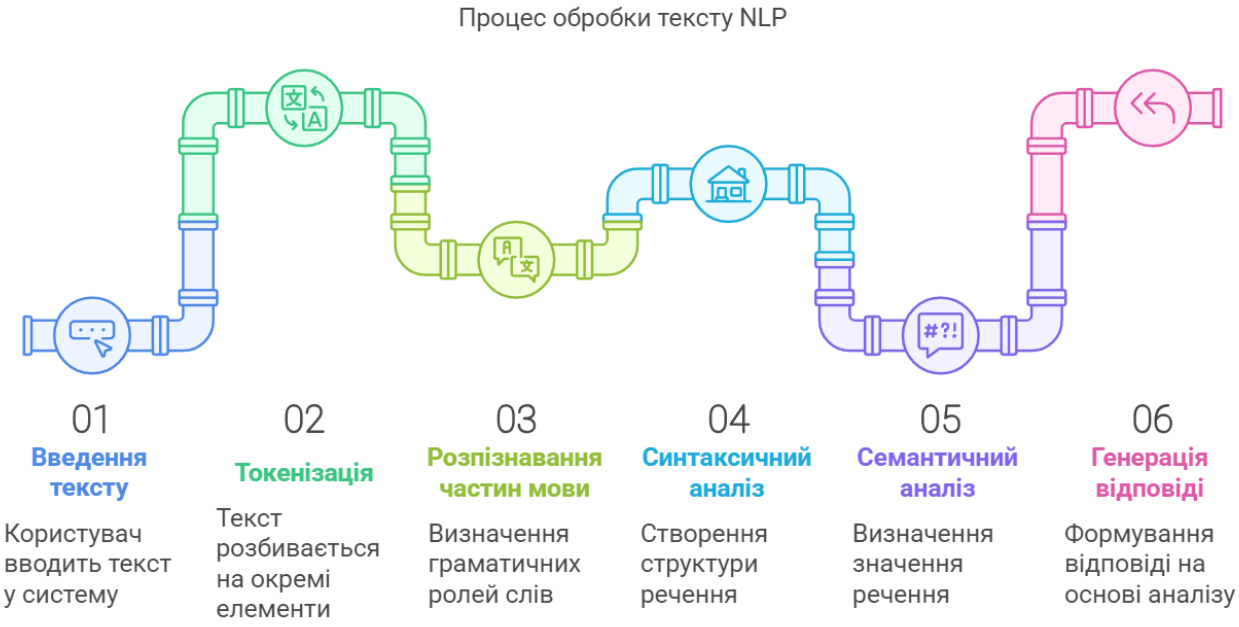

Рисунок 1.3 - Процес обробки тексту за допомогою NLP

Приклади використання NLP у чат-ботах

1. **Amazon Alexa:** Використовує NLP для інтерпретації голосових запитів і виконання команд користувача.
2. **ChatGPT:** Модель, яка генерує текстові відповіді, розуміючи складні контексти та багатозначні запити.
3. **Google Assistant:** Інтегрує NLP для пошуку інформації, встановлення нагадувань і керування розумним будинком.

Аргументи на користь використання NLP у взаємодії з користувачами:

1. **Точність і релевантність:** NLP дозволяє враховувати контекст запиту, що забезпечує релевантність відповідей.
2. **Масштабованість:** Можливість обробляти тисячі запитів одночасно.
3. **Персоналізація:** Аналіз попередніх взаємодій дозволяє створювати індивідуалізовані відповіді.
4. **Адаптивність:** Системи на основі NLP здатні адаптуватися до змін у мові, включаючи нові слова чи сленг.

Проблеми NLP у чат-ботах:

1. **Двозначність:** Інтерпретація багатозначних слів чи виразів може бути складною.
2. **Недостатність даних:** Для ефективного навчання потрібні великі обсяги даних, що не завжди доступно.
3. **Мовні бар'єри:** Використання багатомовних моделей може бути викликом через різні граматичні та лексичні структури мов.

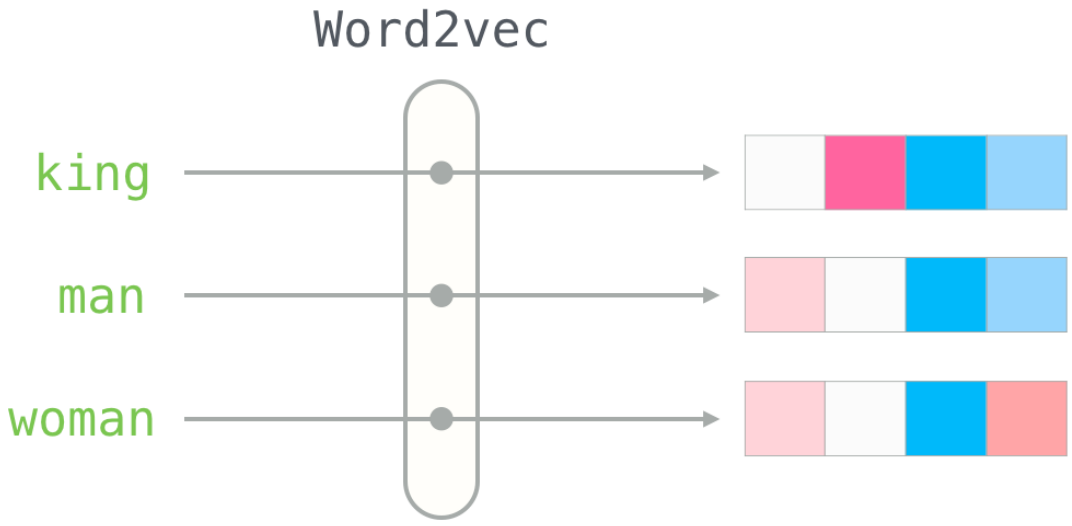

Рисунок 1.4 - Векторизація тексту за допомогою Word2Vec

Обробка природної мови є ключовим елементом у взаємодії чат-ботів із користувачами. Використання NLP дозволяє забезпечити точність, персоналізацію та адаптивність відповідей. Незважаючи на певні виклики, такі як двозначність чи недостатність даних, розвиток NLP продовжує розширювати можливості інтерактивних систем, роблячи їх незамінними в сучасному цифровому середовищі.

1.3 Алгоритми машинного навчання для створення інтерактивних систем

Машинне навчання (Machine Learning, ML) є невід'ємною частиною інтерактивних систем, таких як чат-боти. Використання алгоритмів ML дозволяє створювати системи, здатні адаптуватися до запитів користувачів, аналізувати дані та формувати точні відповіді. У цьому розділі розглянуто ключові алгоритми, що використовуються для створення інтерактивних систем [3].

Основні алгоритми машинного навчання для інтерактивних систем:

1. **Логістична регресія (Logistic Regression):**
    - використовується для задач класифікації, наприклад, визначення категорії запиту (інформаційний, технічний, емоційний тощо);
    - простий у реалізації та ефективний для роботи з невеликими наборами даних;
    - **приклад:** Класифікація запитів користувачів у чат-боті технічної підтримки.
2. **Метод опорних векторів (Support Vector Machine, SVM):**
    - ефективний для задач класифікації та регресії, особливо в багатовимірних просторах;
    - використовується для обробки тексту з високою точністю завдяки здатності створювати нелінійні межі класифікації;
    - **приклад:** Визначення тональності тексту в чат-ботах для аналізу настрою користувачів.
3. **Наївний Баєсів класифікатор (Naive Bayes):**
    - простий та швидкий алгоритм, який часто використовується для класифікації текстів;
    - вимагає невеликої кількості даних для тренування та добре працює з великим обсягом текстової інформації [4];
    - **приклад:** Автоматичне сортування електронних листів (спам чи ні).

4. **Дерева прийняття рішень (Decision Trees):**
    - зручний для візуалізації та інтерпретації процесу прийняття рішень;
    - використовується для побудови сценаріїв взаємодії з користувачами в чат-ботах [5];
    - **приклад:** Побудова структури відповіді на основі вибору користувача в діалоговому меню.
5. **Ансамблеві методи (Random Forest, Gradient Boosting):**
    - поєднують результати кількох моделей для підвищення точності;
    - Random Forest часто використовується для класифікації текстів завдяки стійкості до шуму в даних;
    - **приклад:** Аналіз запитів у багатомовних чат-ботах.
6. **Глибоке навчання (Deep Learning):**
    - використання нейронних мереж для обробки складних даних;
    - архітектури, такі як рекурентні нейронні мережі (RNN) або трансформери (наприклад, BERT [6], GPT [7]), дозволяють враховувати контекст і послідовність у текстах;
    - **приклад:** Генерація відповідей у сучасних чат-ботах (ChatGPT, Google Assistant).

Аргументи на користь використання алгоритмів машинного навчання

1. **Адаптивність:** ML-алгоритми здатні навчатися на основі нових даних, що дозволяє інтерактивним системам постійно вдосконалюватися.
2. **Швидкість і масштабованість:** Використання автоматизованих моделей дозволяє обробляти тисячі запитів одночасно.
3. **Універсальність:** Алгоритми ML можуть бути адаптовані для різних задач — від класифікації тексту до аналізу настрою.

Приклади інтеграції алгоритмів у чат-боти

1. **Аналіз настрою:** Використання SVM для визначення позитивної, негативної чи нейтральної тональності тексту.

2. **Класифікація запитів:** Логістична регресія допомагає розподілити запити за категоріями (пошук інформації, технічна підтримка, скарга).
3. **Генерація відповідей:** Нейронні мережі GPT використовуються для створення релевантних та контекстуальних відповідей.

Проблеми та виклики

1. **Недостатність даних:** Для тренування деяких моделей потрібні великі обсяги високоякісних даних.
2. **Проблема переобучення:** Алгоритми можуть демонструвати високу точність на тренувальних даних, але працювати гірше на реальних запитах.
3. **Час тренування:** Глибокі нейронні мережі потребують значного обсягу обчислювальних ресурсів.

Алгоритми машинного навчання є фундаментом для створення інтерактивних систем, таких як чат-боти. Їх використання дозволяє автоматизувати процеси, підвищити точність аналізу запитів та забезпечити релевантні відповіді. Застосування таких алгоритмів, як SVM, Random Forest та глибокого навчання, відкриває нові горизонти для створення розумних систем, здатних адаптуватися до потреб користувачів і забезпечувати якісний досвід взаємодії.

1.4 Особливості роботи з текстами футбольної тематики

Футбольна тематика є унікальною через специфіку мови, стилю та контексту, які використовуються фанатами, журналістами, тренерами та гравцями. Тексти, пов'язані з футболом, включають новини, коментарі, аналітичні матеріали, обговорення в соціальних мережах, трансляції матчів і багато іншого. Розробка інтерактивних систем, таких як чат-боти для футбольних фанатів, вимагає врахування особливостей цієї тематичної сфери.

Ключові особливості текстів футбольної тематики:

1. **Неструктурованість текстів:**
    - тексти, що генеруються в реальному часі, наприклад коментарі до матчів, часто містять сленг, скорочення, емодзі та специфічні фрази;
    - **приклад:** У коментарях до гри можна зустріти фрази на кшталт "Гол! Суперудар від 10 номера! 🔥" або "Суддя куплений! 😠".
2. **Використання спеціалізованої термінології:**
    - футбольна термінологія включає слова й фрази, такі як "офсайд", "пенальті", "хет-трик", "контратака";
    - ця термінологія має враховуватися при обробці текстів, оскільки неправильне розуміння термінів може спотворити результати аналізу.
3. **Емоційна забарвленість текстів:**
    - футбольні тексти часто мають високу емоційну напругу, оскільки вболівальники активно реагують на події матчу;
    - **приклад:** "Що за жахлива гра! Команда повний провал!" чи "Наші хлопці — герої, найкраща гра сезону!".
4. **Багатомовність:**
    - у контексті міжнародного футболу використовуються тексти різними мовами. Наприклад, новини про Лігу Чемпіонів можуть з'являтися англійською, іспанською, українською тощо;
    - чат-боти повинні враховувати мовні особливості для правильної обробки текстів.
5. **Контекстуальність:**
    - у текстах футбольної тематики часто використовується контекст, зрозумілий лише шанувальникам. Наприклад, згадка "червоні" може означати "Ліверпуль", а "сині" — "Челсі".

Проблеми роботи з текстами футбольної тематики:

1. **Ідентифікація сленгу та жаргону:**
    - сленгові слова та фрази, як-от "класний сейв" або "вийти в євровесну", можуть бути складними для автоматичної інтерпретації.
2. **Обробка емоційно насичених текстів:**
    - аналіз текстів із сильним емоційним забарвленням вимагає використання алгоритмів для визначення тональності та настрою тексту.
3. **Інтеграція футбольних баз даних:**
    - для створення ефективної системи потрібно забезпечити доступ до баз даних, які містять актуальну інформацію про матчі, гравців і турніри.

Підходи до роботи з текстами футбольної тематики:

1. **Попередня обробка тексту:**
    - видалення зайвих символів (емодзі, пунктуації) та слів, які не впливають на зміст;
    - використання токенізації та лематизації для стандартизації тексту.
2. **Використання спеціалізованих словників:**
    - створення словників футбольних термінів для правильного розпізнавання специфічних слів і фраз.
3. **Аналіз настрою:**
    - використання алгоритмів для класифікації текстів за емоційним забарвленням (позитивне, негативне, нейтральне).
4. **Машинне навчання:**
    - навчання моделей на основі реальних текстів з футбольних форумів, соціальних мереж і новин;
    - використання Word2Vec або BERT для врахування контекстуальних залежностей у текстах.

Приклади використання:

- чат-боти клубів, таких як "Барселона" чи "Манчестер Юнайтед", надають актуальну інформацію про матчі, статистику та новини, враховуючи специфіку мови фанатів;
- аналітичні платформи, які визначають настрій фанатів перед і після матчів, використовуючи NLP для аналізу соціальних мереж.

Робота з текстами футбольної тематики вимагає врахування багатьох специфічних особливостей, таких як сленг, емоційна насиченість і контекстуальність. Використання сучасних методів обробки природної мови та алгоритмів машинного навчання дозволяє створювати ефективні системи для аналізу текстів і взаємодії з футбольними фанатами. Це відкриває перспективи для покращення досвіду користувачів і розширення функціональності інтерактивних платформ.

1.5 Висновки до Розділу 1

У першому розділі було досліджено теоретичні основи, які є базисом для розробки інтерактивних систем, зокрема чат-ботів, що використовують технології обробки природної мови (NLP) і машинного навчання (ML). Проведений аналіз дозволив визначити основні аспекти, які слід враховувати при розробці системи для взаємодії з футбольними фанатами.

Ключові висновки:

1. **Визначення ролі чат-ботів у цифрових комунікаціях.** Чат-боти є важливим інструментом для автоматизації взаємодії з користувачами, що забезпечує швидке та ефективне обслуговування. Їх інтерактивність, гнучкість і масштабованість дозволяють обробляти велику кількість запитів у реальному часі. Наприклад, футбольні клуби можуть використовувати чат-боти для надання інформації про матчі, статистику гравців або продаж квитків.

2. **Роль NLP у взаємодії з користувачами.** Технології обробки природної мови є основою для розуміння та аналізу текстів, які генерують користувачі. Використання NLP дозволяє обробляти текстові дані, враховуючи контекст, емоційне забарвлення та специфічну лексику. Наприклад, аналіз текстів футбольної тематики потребує виявлення сленгу, спеціалізованої термінології та емоційного фону.
3. **Алгоритми машинного навчання для обробки текстів.** Сучасні алгоритми ML, такі як SVM, Random Forest та нейронні мережі, забезпечують високу точність класифікації та аналізу текстів. Для інтерактивних систем вони є ключовими компонентами, які дозволяють створювати персоналізовані відповіді та забезпечувати адаптивність до змін у даних. Наприклад, трансформери (GPT, BERT) використовуються для генерації контекстуальних відповідей у чат-ботах.
4. **Особливості текстів футбольної тематики.** Футбольні тексти є унікальними через їхню емоційність, використання спеціалізованої термінології та неструктурованість. Для ефективної роботи з ними слід використовувати спеціалізовані словники, алгоритми аналізу настроїв і методи попередньої обробки текстів. Наприклад, у соціальних мережах фанати активно використовують сленг та скорочення, що ускладнює автоматичний аналіз без попередньої стандартизації тексту.
5. **Взаємозв'язок між теоретичними аспектами та практичною реалізацією.** Теоретичні аспекти, розглянуті у цьому розділі, створюють фундамент для практичної реалізації чат-бота для футбольних фанатів. Використання алгоритмів машинного навчання та NLP дозволяє створити інтерактивну систему, яка забезпечить точну та релевантну взаємодію з користувачами.

Практичне значення:

Вивчення цих аспектів дозволяє:

- забезпечити високу точність аналізу текстів користувачів, що є важливим для чат-ботів у спортивній індустрії;
- створити систему, яка враховує специфіку футбольної лексики та контексту;
- забезпечити масштабованість і адаптивність чат-бота для роботи з великими аудиторіями.

Проведений аналіз теоретичних основ підкреслює важливість комплексного підходу до розробки інтерактивних систем. Використання сучасних методів NLP та ML у поєднанні з урахуванням специфіки текстів футбольної тематики є ключем до створення ефективних чат-ботів, які зможуть задовольнити потреби користувачів у спортивній індустрії.

РОЗДІЛ 2
МЕТОДИКА ТА ЗАСОБИ РОЗРОБКИ ЧАТ-БОТА ДЛЯ ФУТБОЛЬНИХ ФАНАТІВ

2.1 Постановка задачі та формулювання вимог до системи

Постановка задачі

Розробка інтерактивного чат-бота для футбольних фанатів вимагає створення системи, яка зможе забезпечити швидке, зручне та точне надання інформації, адаптованої до потреб користувачів. Основною задачею є інтеграція обробки природної мови (NLP) та алгоритмів машинного навчання для ефективної обробки текстових запитів, взаємодії з футбольними базами даних і формування релевантних відповідей.

Система повинна виконувати наступні ключові функції:

1. Аналіз текстових запитів користувачів для визначення їхньої суті та наміру (intent detection).
2. Надання інформації про футбольні матчі, команди, гравців, турнірну таблицю та інші актуальні події.
3. Інтерактивна взаємодія з користувачами, включаючи відповіді на запити, аналіз настрою текстів і персоналізацію відповідей.
4. Інтеграція з футбольними базами даних для отримання актуальної інформації.
5. Забезпечення роботи системи в реальному часі з можливістю обробки великої кількості запитів одночасно.

Формулювання вимог до системи:

Функціональні вимоги:

1. **Аналіз текстових запитів:**
    - система повинна розуміти запити користувачів, написані природною мовою, включаючи сленг і скорочення;

- приклад: Запит "Коли грає Барселона?" повинен коректно інтерпретуватися і повертати дату наступного матчу команди.

2. **Інформаційне забезпечення:**
    - чат-бот повинен надавати актуальну інформацію про футбольні матчі, статистику гравців, турнірні таблиці, результати ігор та інші дані;
    - інтеграція з базами даних, такими як API спортивних сервісів (наприклад, FootballData, Sportradar).
3. **Обробка емоційного забарвлення:**
    - система повинна аналізувати емоційний стан тексту користувача для формування відповідного тону відповіді;
    - приклад: На негативний запит "Чому наша команда програла?" бот повинен відповідати в нейтральному тоні з поясненням результатів гри.
4. **Персоналізація:**
    - чат-бот має запам'ятовувати попередні взаємодії з користувачем і формувати відповіді на основі цієї інформації;
    - наприклад, якщо користувач часто питає про "Манчестер Юнайтед", бот може пропонувати актуальні новини про цю команду.
5. **Інтерактивність:**
    - система повинна підтримувати інтерактивний діалог, дозволяючи користувачам уточнювати свої запити або переходити до суміжних тем.

Нефункціональні вимоги:

1. **Продуктивність:**
    - чат-бот повинен обробляти запити в режимі реального часу з часом відповіді не більше 1-2 секунд.

2. **Масштабованість:**
    - система повинна підтримувати одночасну роботу з великою кількістю користувачів (не менше 10,000 активних сесій).
3. **Надійність:**
    - забезпечення високої доступності системи (≥ 99% uptime).
4. **Захист даних:**
    - гарантування конфіденційності даних користувачів відповідно до стандартів GDPR.
5. **Кросплатформеність:**
    - чат-бот має працювати в різних месенджерах (Telegram, Facebook Messenger, WhatsApp) і бути доступним через веб-інтерфейс.

Архітектура системи

Система складається з наступних компонентів:

1. **Модуль NLP:** Використовується для обробки текстових запитів (токенізація, лематизація, класифікація намірів).
2. **Модуль машинного навчання:** Здійснює аналіз текстів, включаючи аналіз настрою та класифікацію.
3. **База знань:** Містить футбольну інформацію, яка постійно оновлюється через API.
4. **Інтерфейс користувача:** Взаємодія з користувачем через месенджери та веб-додатки.
5. **Серверна частина:** Забезпечує обробку запитів та інтеграцію компонентів системи.

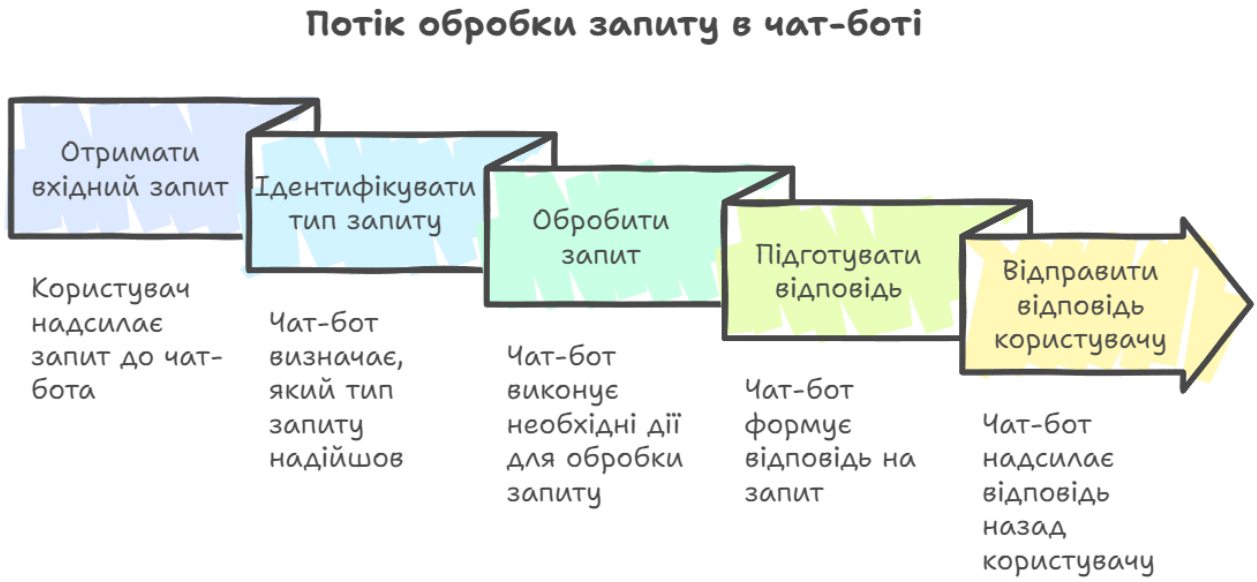

Рисунок 2.1 - Потік обробки запиту в чат-боті

Постановка задачі та формулювання вимог є ключовими етапами при розробці чат-бота для футбольних фанатів. Забезпечення відповідності функціональних і нефункціональних вимог дозволяє створити систему, яка задовольняє потреби користувачів, працює надійно і ефективно. Розроблена архітектура забезпечує інтеграцію NLP, машинного навчання та баз знань, що є основою для реалізації системи.

2.2 Вибір методів обробки текстових даних

Обробка текстових даних є ключовим етапом у розробці інтерактивних систем, таких як чат-боти. Правильний вибір методів обробки тексту забезпечує точність аналізу та релевантність відповідей. У цьому підрозділі розглянуто основні методи обробки тексту, які використовуються для аналізу запитів користувачів і генерації відповідей у чат-боті для футбольних фанатів.

Основні етапи обробки тексту:

1. **Попередня обробка тексту.** Попередня обробка спрямована на підготовку тексту до подальшого аналізу, зокрема:
    - **очищення тексту:** Видалення зайвих символів (пунктуації, емодзі), чисел і HTML-тегів;
    - **видалення стоп-слів:** Усунення загальних слів ("і", "або", "але"), які не впливають на зміст;
    - **токенізація:** Розбиття тексту на окремі слова чи фрази (токени);
    - **лематизація:** Приведення слів до їх базової форми ("гравець" замість "гравці");
    - **нормалізація:** Приведення тексту до нижнього регістру для забезпечення уніфікованості даних.

      Приклад: Запит "Коли грає Барселона?" після обробки стає: "коли грати барселона".
2. **Векторизація тексту.** Для використання текстових даних у алгоритмах машинного навчання їх потрібно перетворити в числовий формат. Основні методи векторизації [8]:
    - **Bag of Words (BoW):** Представлення тексту у вигляді вектора, де кожна позиція відповідає частоті появи певного слова [9];
    - **TF-IDF (Term Frequency-Inverse Document Frequency):** Враховує частоту слова в документі та його важливість у всьому корпусі текстів [10];
    - **Word Embeddings:** Використання моделей, таких як Word2Vec [11] або GloVe [12], для створення векторів, які зберігають семантичні зв’язки між словами.

Переваги TF-IDF:

- враховує унікальність слів для кожного документа;
- зменшує вагу часто вживаних слів, таких як "матч", "команда".

Приклад: Слово "гол" у футбольному тексті матиме високий ваговий коефіцієнт у TF-IDF, якщо воно зустрічається рідко, але є ключовим для даного документа.

3. **Аналіз настрою (Sentiment Analysis).** Аналіз настрою використовується для визначення емоційного забарвлення тексту (позитивне, негативне, нейтральне). Наприклад:
    - запит "Це була жахлива гра!" має негативний настрій;
    - запит "Наша команда грала чудово!" — позитивний.

Для аналізу настрою часто використовуються алгоритми SVM або нейронні мережі, які навчаються на анотованих текстових даних.

4. **Виділення сутностей (Named Entity Recognition, NER).** Цей метод використовується для ідентифікації ключових об’єктів у тексті, таких як:
    - імена гравців ("Мессі", "Роналду");
    - назви команд ("Барселона", "Челсі");
    - дати та час ("15 квітня", "о 19:00").

Приклад: Запит "Коли грає Реал Мадрид?" — NER виділяє об’єкт "Реал Мадрид" і дату матчу.

5. **Класифікація тексту.** Цей етап включає визначення категорії запиту користувача, наприклад:
    - інформаційний запит ("Коли наступний матч?");
    - емоційний запит ("Чому ми програли?").

Для класифікації текстів часто використовуються алгоритми, такі як логістична регресія, наївний Баєс або трансформери (BERT).

Вибір оптимальних методів для чат-бота футбольної тематики:

Для системи, орієнтованої на футбольних фанатів, найкраще підходять:

1. **TF-IDF для векторизації тексту:** забезпечує ефективність обробки великого обсягу текстів.
2. **NER для виділення сутностей:** дозволяє коректно ідентифікувати команди, гравців і дати.

3. **Аналіз настрою:** допомагає адаптувати відповіді відповідно до емоційного стану користувача.
4. **Токенізація та лематизація:** для підготовки тексту до аналізу, враховуючи специфіку футбольної лексики.

Проблеми та виклики:

1. **Сленг і скорочення:** Використання фанатами специфічних виразів, таких як "ГГ" ("гарна гра"), ускладнює обробку текстів.
2. **Багатомовність:** Запити можуть бути написані різними мовами, наприклад, українською, англійською чи іспанською.
3. **Контекстуальність:** Одні й ті самі слова можуть мати різний сенс залежно від контексту ("червоні" — "Ліверпуль" чи "Манчестер Юнайтед").

Вибір методів обробки текстових даних є критичним для забезпечення точності та ефективності роботи чат-бота. Використання токенізації, лематизації, TF-IDF, NER та аналізу настрою дозволяє створити систему, яка коректно інтерпретує запити користувачів, враховує специфіку футбольної тематики та забезпечує високу якість взаємодії. Подальша оптимізація цих методів, включаючи адаптацію до сленгу та багатомовності, є ключовою для покращення функціональності системи.

2.3 Архітектура системи чат-бота

Архітектура чат-бота для футбольних фанатів повинна забезпечувати ефективну інтерактивну взаємодію з користувачами, враховувати специфіку текстових запитів та обробляти великий обсяг даних у режимі реального часу. У цьому підрозділі представлено основні компоненти архітектури системи, їх функції та взаємодію між ними.

Основні компоненти архітектури системи:

1. **Модуль обробки природної мови (NLP):**
    - виконує попередню обробку тексту (токенізацію, лематизацію, видалення стоп-слів);
    - класифікує наміри користувача (intent detection) та виділяє ключові сутності (NER);
    - **приклад:** Запит "Коли наступний матч Барселони?" розпізнається як запит інформації про дату матчу, а "Барселона" визначається як ключова сутність.
2. **Модуль машинного навчання (ML):**
    - використовується для аналізу настрою текстів, класифікації запитів та створення персоналізованих рекомендацій;
    - **алгоритми:**
        - логістична регресія — для класифікації намірів;
        - Random Forest — для виявлення важливих елементів у тексті;
        - GPT або BERT — для генерації відповідей на основі контексту.
3. **База знань:**
    - містить структуровану інформацію про футбольні команди, матчі, результати, гравців та статистику;
    - підтримує оновлення даних через інтеграцію з API (наприклад, FootballData, Sportradar);
    - **приклад:** Для запиту "Хто забив найбільше голів у сезоні?" база знань надає відповідну статистику.
4. **Модуль генерації відповідей:**
    - формує відповіді на основі аналізу запиту користувача та даних із бази знань;
    - забезпечує релевантність і природність тексту;
    - **приклад:** На запит "Який рахунок у матчі Реал-Мадрид?" генерується відповідь: "Реал переміг із рахунком 3:1".

5. **Інтерфейс користувача:**
    - забезпечує взаємодію між користувачем та системою через різні платформи (месенджери, веб-додатки);
    - інтегрується з Telegram, Facebook Messenger, WhatsApp тощо;
    - **особливості:**
        - простий у використанні;
        - підтримка мультимовності.
6. **Серверна частина:**
    - управляє обробкою запитів, взаємодією між компонентами системи та зберіганням даних;
    - використовує REST API для обміну даними між модулями.

Потік обробки запиту:

1. Користувач надсилає запит через месенджер або веб-інтерфейс.
2. Інтерфейс передає запит серверу.
3. Сервер надсилає текст у модуль NLP для попередньої обробки та класифікації.
4. Модуль ML аналізує запит і визначає відповідний контекст або настрій.
5. База знань надає необхідну інформацію.
6. Модуль генерації формує відповідь і повертає її користувачеві.

Аргументи на користь вибраної архітектури:

1. **Модульність:** Кожен компонент виконує чітко визначену функцію, що забезпечує легкість масштабування та оновлення.
2. **Гнучкість:** Можливість адаптувати систему до різних мов, сленгу та потреб користувачів.
3. **Швидкість:** Обробка запитів у режимі реального часу забезпечує високу продуктивність.
4. **Масштабованість:** Система підтримує одночасну взаємодію з тисячами користувачів.

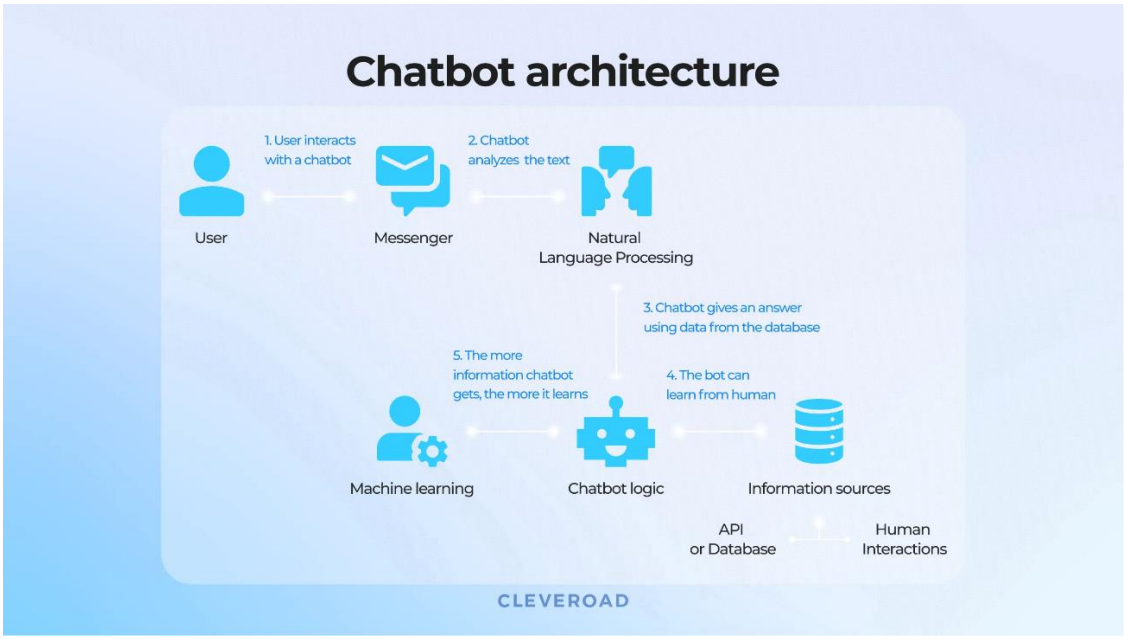

Рисунок 2.2 - Архітектура системи чат-бота

Архітектура системи чат-бота для футбольних фанатів забезпечує ефективну обробку текстових запитів, інтеграцію з футбольними базами даних і генерацію релевантних відповідей. Її модульність, масштабованість і здатність адаптуватися до потреб користувачів роблять систему універсальним інструментом для взаємодії з фанатами в цифровому середовищі.

2.4 Інтеграція з футбольними інформаційними ресурсами

Інтеграція з футбольними інформаційними ресурсами є одним із ключових аспектів для забезпечення актуальності та точності відповідей чат-бота. Доступ до даних про команди, матчі, гравців і статистику дозволяє створити ефективну інформаційну систему, яка задовольняє потреби користувачів.

Основні етапи інтеграції:

1. **Вибір джерел даних.** Для забезпечення повноти та актуальності інформації використовуються наступні джерела:
    - **API сторонніх сервісів:**
        - **FootballData API:** надає дані про результати матчів, турнірні таблиці та розклад ігор;
        - **Sportradar:** пропонує розширену статистику гравців, команд і ліг.
    - **офіційні вебсайти ліг та клубів:** використовуються для перевірки точності інформації та доступу до ексклюзивних даних;
    - **соціальні мережі:** дозволяють відстежувати оперативну інформацію про події.
2. **Інтеграція через API:**
    - **REST API:** найбільш поширений спосіб інтеграції. Він дозволяє здійснювати запити до сторонніх сервісів і отримувати структуровані відповіді у форматі JSON або XML;
    - **вебхуки:** використовуються для отримання реального часу оновлень, наприклад, під час трансляції матчів.

    Приклад: Запит до FootballData API:

    GET https://api.football-data.org/v2/competitions/CL/matches

    Headers: { "Authorization": "Bearer {API_KEY}" }

    Відповідь містить розклад матчів Ліги Чемпіонів у JSON-форматі.
3. **Створення локальної бази даних.** Для зменшення часу відповіді та забезпечення стабільної роботи системи створюється локальна база даних, яка синхронізується з API. Вона зберігає такі дані:
    - інформацію про команди (назва, гравці, тренер, емблема);
    - результати матчів (дата, рахунок, події);
    - турнірні таблиці.

4. **Автоматизація оновлень:**
    - налаштовуються періодичні запити до API для оновлення бази даних;
    - використовуються тригери для оновлення даних у режимі реального часу;

Аргументи на користь інтеграції:

1. **Актуальність інформації:** Інтеграція з офіційними ресурсами гарантує, що користувачі отримують найсвіжіші дані.
2. **Швидкість відповіді:** Локальна база даних дозволяє зменшити затримки під час отримання відповідей.
3. **Масштабованість:** Використання API забезпечує легке додавання нових ліг чи турнірів.
4. **Гнучкість:** Можливість підключення до кількох джерел даних одночасно.

Приклади використання інтеграції:

- **оновлення турнірних таблиць:** Користувач запитує: "Яке місце займає Манчестер Юнайтед у Прем'єр-лізі?" Чат-бот звертається до бази даних, оновленої через API, і надає відповідь;
- **розклад матчів:** На запит "Коли грає Барселона?" система надає дату й час матчу, отримані з FootballData API;
- **статистика гравців:** Користувач запитує: "Скільки голів забив Мессі у цьому сезоні?" Чат-бот звертається до локальної бази даних, синхронізованої з Sportradar.

Проблеми інтеграції:

1. **Обмеження API:** Деякі сервіси мають ліміти на кількість запитів, що може вимагати оптимізації.
2. **Несумісність форматів:** Дані з різних джерел можуть мати різний формат, що потребує додаткової обробки.
3. **Робота в реальному часі:** Під час матчів дані змінюються динамічно, що вимагає ефективних алгоритмів синхронізації.

Інтеграція з футбольними інформаційними ресурсами забезпечує актуальність, точність і швидкість надання відповідей у чат-боті. Використання API та локальної бази даних дозволяє створити ефективну систему, яка задовольняє потреби футбольних фанатів і надає релевантну інформацію в режимі реального часу. Подальша оптимізація цього процесу сприятиме підвищенню продуктивності системи.

2.5 Інструменти для розробки та тестування

Розробка та тестування чат-бота для футбольних фанатів вимагає використання спеціалізованих інструментів, які забезпечують ефективність створення, налагодження та перевірки функціональності системи. У цьому розділі розглянуто ключові інструменти для кожного етапу розробки.

Інструменти для розробки:

1. **Фреймворки для створення чат-ботів:**
    - **Rasa** [13]**:**
        - пропонує відкритий код для розробки NLP-підсистем та інтеграції з месенджерами;
        - переваги: гнучкість, підтримка кастомізації моделей, інтеграція з базами даних;
        - **приклад:** Використання Rasa для класифікації намірів користувача та виявлення сутностей.
    - **Dialogflow (Google):**
        - інструмент для швидкого створення діалогів з підтримкою NLP;
        - переваги: простий інтерфейс, автоматична інтеграція з Google Assistant.

- **Flask** [14]**:**
    - інструмент для створення веб-додатків;
    - переваги: простий та зручний у використанні.

2. **Мови програмування:**
    - **Python:**
        - основна мова для створення NLP-моделей та інтеграції з API;
        - бібліотеки: NLTK, SpaCy, TensorFlow, PyTorch;
        - **приклад:** Використання TensorFlow для навчання моделі аналізу настрою [15].
    - **JavaScript:**
        - для розробки фронтенду чат-бота або інтеграції з платформами;
        - бібліотеки: Botpress, Microsoft Bot Framework.
3. **Системи керування базами даних (DBMS):**
    - **PostgreSQL:**
        - реляційна база даних для зберігання структурованої інформації про матчі, команди та результати.
    - **MongoDB:**
        - документоорієнтована база даних для зберігання неструктурованих текстів;
        - **приклад:** Збереження текстових запитів користувачів для подальшого аналізу.
4. **Інструменти інтеграції з API:**
    - **Postman:**
        - використовується для тестування запитів до сторонніх API (наприклад, FootballData API);
        - **приклад:** Перевірка роботи REST-запитів для отримання інформації про матчі.

- **cURL:**
    - консольний інструмент для швидкого виконання HTTP-запитів.

Інструменти для тестування:

1. **Середовища для автоматичного тестування:**
    - **Pytest:**
        - використовується для написання та виконання модульних тестів;
        - **приклад:** Тестування функцій токенізації та класифікації запитів.
    - **Jest (JavaScript):**
        - для тестування фронтенду чат-бота.
2. **Тестування навантаження:**
    - **Apache JMeter:**
        - аналіз продуктивності системи під час обробки великої кількості запитів;
        - **приклад:** Визначення максимальної кількості одночасних користувачів.
    - **Locust:**
        - для моделювання поведінки користувачів у реальному часі.
3. **Інструменти моніторингу та налагодження:**
    - **Elasticsearch + Kibana:**
        - моніторинг запитів та візуалізація роботи системи в реальному часі.
    - **Sentry:**
        - відстеження помилок у роботі бота та швидке їх виправлення.

Аргументи на користь використання інструментів:

1. **Продуктивність:** Інструменти, такі як Rasa чи TensorFlow, дозволяють швидко створювати високоефективні NLP-моделі.

2. **Масштабованість:** Використання PostgreSQL чи MongoDB забезпечує легке розширення баз даних.
3. **Надійність:** Тестування через Pytest і JMeter дозволяє гарантувати стабільну роботу системи.
4. **Швидкість розробки:** Інструменти типу Postman спрощують інтеграцію з API.

Приклади використання інструментів

- **розробка NLP-моделі:** Використання SpaCy для токенізації запитів та NLTK для аналізу настрою текстів;
- **тестування продуктивності:** Locust моделює поведінку 10,000 користувачів, які одночасно надсилають запити до чат-бота;
- **моніторинг:** Sentry автоматично повідомляє про помилки в роботі бота під час обробки реальних запитів.

Рисунок 2.3 – Цикл розробки чат-бота

Використання сучасних інструментів для розробки та тестування дозволяє створити надійну, масштабовану та ефективну систему чат-бота. Комбінація фреймворків, мов програмування, баз даних і тестувальних середовищ забезпечує комплексний підхід до створення продукту, який відповідає потребам користувачів і вимогам сучасної індустрії.

2.6 Висновки до Розділу 2

У другому розділі було розглянуто ключові аспекти, які визначають структуру, функціональність та інструменти для розробки чат-бота для футбольних фанатів. Проведений аналіз дозволив сформулювати важливі висновки щодо постановки задачі, вибору методів обробки тексту, архітектури системи, інтеграції з футбольними інформаційними ресурсами та використання відповідних інструментів.

Ключові висновки:

1. **Постановка задачі та формулювання вимог.** Чітке визначення задачі дозволило сформулювати функціональні та нефункціональні вимоги до системи. Основними потребами є:
    - аналіз текстових запитів користувачів для визначення намірів та ключових сутностей;
    - забезпечення актуальної інформації про матчі, команди та статистику;
    - підтримка інтерактивної взаємодії в реальному часі. Прикладом є можливість надання користувачеві розкладу матчів у запиті: "Коли грає Барселона?".
2. **Методи обробки текстових даних.** Вибір методів обробки тексту, таких як токенізація, лематизація, TF-IDF та NER, забезпечує високу точність розуміння запитів користувачів. Використання аналізу настрою додає можливість адаптувати відповіді залежно від емоційного стану

користувача. Наприклад, позитивний запит "Наша команда найкраща!" обробляється відповідним чином.

3. **Архітектура системи.** Модульна архітектура системи дозволяє забезпечити гнучкість, масштабованість та інтеграцію з різними платформами. Основні компоненти, такі як модуль NLP, база знань та модуль генерації відповідей, забезпечують ефективну роботу системи. Наприклад, модуль NLP виконує попередню обробку тексту, а база знань надає актуальні дані для формування відповідей.
4. **Інтеграція з футбольними інформаційними ресурсами.** Інтеграція через REST API з такими сервісами, як FootballData API, дозволяє надавати актуальну інформацію про футбольні події. Створення локальної бази даних зменшує час відповіді та забезпечує стабільність роботи системи навіть при високому навантаженні. Наприклад, запит "Яке місце займає Манчестер Юнайтед у таблиці?" обробляється через синхронізовану базу даних.
5. **Інструменти для розробки та тестування.** Використання таких інструментів, як Rasa для розробки NLP-моделей, PostgreSQL для керування базами даних та JMeter для тестування навантаження, забезпечує високу продуктивність та надійність системи. Наприклад, JMeter дозволяє протестувати продуктивність системи при одночасному надсиланні тисяч запитів.

Аргументи на користь запропонованого підходу:

- **модульність:** Чітка структура системи дозволяє легко впроваджувати зміни та оновлення;
- **продуктивність:** Використання сучасних інструментів та методів забезпечує високу швидкість обробки запитів;
- **гнучкість:** Система легко адаптується до нових потреб користувачів та змін у футбольній індустрії;
- **актуальність:** Інтеграція з офіційними ресурсами гарантує достовірність даних.

Рекомендації для подальших досліджень:

1. Впровадження мультимовних моделей NLP для розширення аудиторії.
2. Використання глибоких нейронних мереж для покращення генерації відповідей.
3. Розробка додаткових функцій, таких як персоналізовані сповіщення про матчі.

Проведений аналіз і вибір рішень у другому розділі забезпечують міцний фундамент для розробки ефективної системи чат-бота. Використання сучасних методів обробки тексту, модульної архітектури та інтеграції з інформаційними ресурсами дозволяє створити надійний інструмент, який задовольнить потреби футбольних фанатів у зручному доступі до актуальної інформації.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

3.1 Етапи реалізації функціоналу чат-бота

Реалізація функціоналу чат-бота включає кілька ключових етапів, які забезпечують повну інтерактивність, точність відповідей і зручність у використанні. У цьому підрозділі розглянуто основні етапи розробки та впровадження чат-бота для футбольних фанатів.

1. Планування та визначення вимог:

На початковому етапі здійснюється аналіз вимог до системи та її функціональних можливостей. Основні завдання:

- визначення функцій, таких як пошук інформації про матчі, статистику гравців і турнірні таблиці;
- формулювання нефункціональних вимог (швидкість обробки запитів, масштабованість, інтеграція з API);
- обговорення платформ для розгортання, зокрема Telegram, WhatsApp, Facebook Messenger.

2. Розробка архітектури системи:

Архітектура базується на модульному підході:

- модуль NLP обробляє запити користувачів, визначає наміри (intent detection) та сутності (entity recognition);
- модуль інтеграції з API отримує інформацію з футбольних баз даних;
- логіка відповідей генерується на основі оброблених запитів і актуальних даних;
- використання **Flask** як фреймворку для створення веб-додатка забезпечує легкість розгортання та інтеграції компонентів.

Чому Flask?

- Flask є легким і гнучким веб-фреймворком, який дозволяє швидко створювати RESTful API;
- простота інтеграції з бібліотеками NLP, такими як Rasa чи SpaCy;
- можливість масштабування завдяки підтримці розширень.

3. Реалізація NLP-функціоналу:

Основні кроки включають:

- використання бібліотек **SpaCy** та **NLTK** для токенізації, лематизації та видалення стоп-слів;
- навчання моделей для класифікації намірів користувачів;
- використання Named Entity Recognition (NER) для виділення ключових сутностей, таких як назви команд чи дати.

4. Інтеграція з базами даних і API:

Для забезпечення актуальності інформації інтегруються такі інструменти:

- **FootballData API**: забезпечує дані про матчі, турнірні таблиці та гравців;
- **PostgreSQL** для зберігання структурованих даних локально;
- реалізація кешування для зменшення навантаження на API та прискорення відповідей.

5. Реалізація модуля генерації відповідей:

Модуль формує відповіді на основі оброблених даних:

- генерація текстів із використанням GPT-моделей;
- формування коротких відповідей для типових запитів, наприклад: "Коли грає Барселона?" — "Наступний матч 10 січня о 19:00."

6. Тестування функціональності:

Тестування забезпечує стабільність роботи:

- **модульне тестування** кожного компонента за допомогою Pytest;
- **тестування навантаження** через JMeter для моделювання 10,000 одночасних запитів;
- **ручне тестування** взаємодії в месенджерах.

7. Розгортання та інтеграція з платформами:

Після завершення розробки здійснюється:

- налаштування серверів для розгортання веб-додатка на Flask;
- інтеграція з Telegram через Telegram Bot API;
- забезпечення доступності чат-бота через веб-інтерфейс для додаткової зручності.

Додаткові аспекти:

- використання Docker для контейнеризації компонентів системи;
- моніторинг через Elasticsearch і Kibana для відстеження продуктивності.

Етапи реалізації функціоналу чат-бота охоплюють весь цикл розробки — від планування до інтеграції з платформами. Використання Flask як веб-фреймворку забезпечує легкість розгортання та масштабованість системи. Сучасні інструменти для NLP, тестування та інтеграції роблять чат-бот ефективним і зручним для взаємодії з футбольними фанатами.

3.2 Підготовка текстових даних до обробки

Підготовка текстових даних є критично важливим етапом у створенні системи обробки природної мови (NLP). Вона включає попереднє очищення, перетворення та структурування тексту для забезпечення ефективної роботи алгоритмів машинного навчання. У цьому підрозділі детально розглянуто етапи підготовки текстових даних для чат-бота футбольної тематики.

1. Очищення тексту:

Текстові дані можуть містити багато шуму, такого як непотрібні символи, числа, HTML-теги чи емодзі. Очищення тексту забезпечує його стандартизацію.

- **видалення зайвих символів:** Видаляються знаки пунктуації, спеціальні символи, хештеги, емодзі та HTML-теги:
  - **приклад:** Текст "Гол! ⚽ Барселона перемогла!" після очищення перетворюється на "Гол Барселона перемогла".

- **обробка чисел:** Числа можуть видалятися або замінюватися залежно від завдання. Наприклад, у футбольних текстах результат матчу ("3:1") може залишатися.

2. Нормалізація тексту:

Нормалізація приводить текст до уніфікованого вигляду:

- **приведення до нижнього регістру:** Всі слова перетворюються на малі літери для уникнення дублювання:
  - **приклад:** "Барселона" та "барселона" обробляються як одне слово.
- **розширення скорочень:** Замінюються скорочення на повні слова (наприклад, "топ-гравець" -> "топ гравець").

3. Видалення стоп-слів:

Стоп-слова — це слова, які не несуть змістовного навантаження ("і", "але", "також"). Вони видаляються для зменшення обсягу тексту та фокусування на ключових термінах.

- для цього використовуються спеціалізовані списки стоп-слів, які можуть адаптуватися до футбольної тематики:
  - **приклад:** "Команда грала добре, але програла" перетворюється на "Команда грала програла".

4. Токенізація:

Токенізація розбиває текст на окремі слова, речення або фрази (токени):

- **словесна токенізація:** Розділення тексту на окремі слова;
- **реченнєва токенізація:** Розбиття тексту на речення:
  - **приклад:** "Барселона грала з Реалом. Рахунок 2:1." Токенізується як: ["Барселона", "грала", "з", "Реалом", "Рахунок", "2", "1"].

5. Лематизація та стемінг:

Ці процеси зводять слова до їхньої базової форми:

- **лематизація:** Використання словника для визначення базової форми слова (наприклад, "грала" -> "грати");

- **стемінг:** Видалення закінчень слів без урахування граматики (наприклад, "грала" -> "грал").

  Перевага лематизації: Більш точна, ніж стемінг, особливо для складної граматики.

6. Обробка спеціалізованих футбольних термінів:

Футбольні тексти містять багато спеціалізованої термінології ("хет-трик", "офсайд", "пенальті").

- для цього створюється словник футбольних термінів;
- **приклад:** Термін "хет-трик" зберігається у первісному вигляді, щоб забезпечити правильну інтерпретацію.

7. Аналіз настрою тексту (Sentiment Analysis):

Для взаємодії з користувачами важливо враховувати емоційне забарвлення тексту:

- **позитивні:** "Наша команда перемогла!";
- **негативні:** "Гра була жахлива.";
- використовуються попередньо навчені моделі для визначення настрою.

8. Векторизація тексту:

Для використання тексту в алгоритмах машинного навчання його потрібно перетворити в числовий формат:

- **Bag of Words (BoW):** Модель, яка враховує частоту слів у тексті [11];
- **TF-IDF (Term Frequency-Inverse Document Frequency):** Визначає важливість слова у тексті відносно всього корпусу текстів [12];
- **Word Embeddings (Word2Vec** [13]**, GloVe** [14]**):** Створює вектори, які зберігають семантичні зв’язки між словами.

9. Підготовка даних для мультимовного аналізу:

Футбольна тематика часто охоплює кілька мов (наприклад, українська, англійська, іспанська). Для цього:

- використовуються мультимовні моделі, такі як BERT;
- забезпечується коректне кодування тексту (UTF-8).

Рисунок 3.1 - Процес токенізації та лематизації футбольного тексту

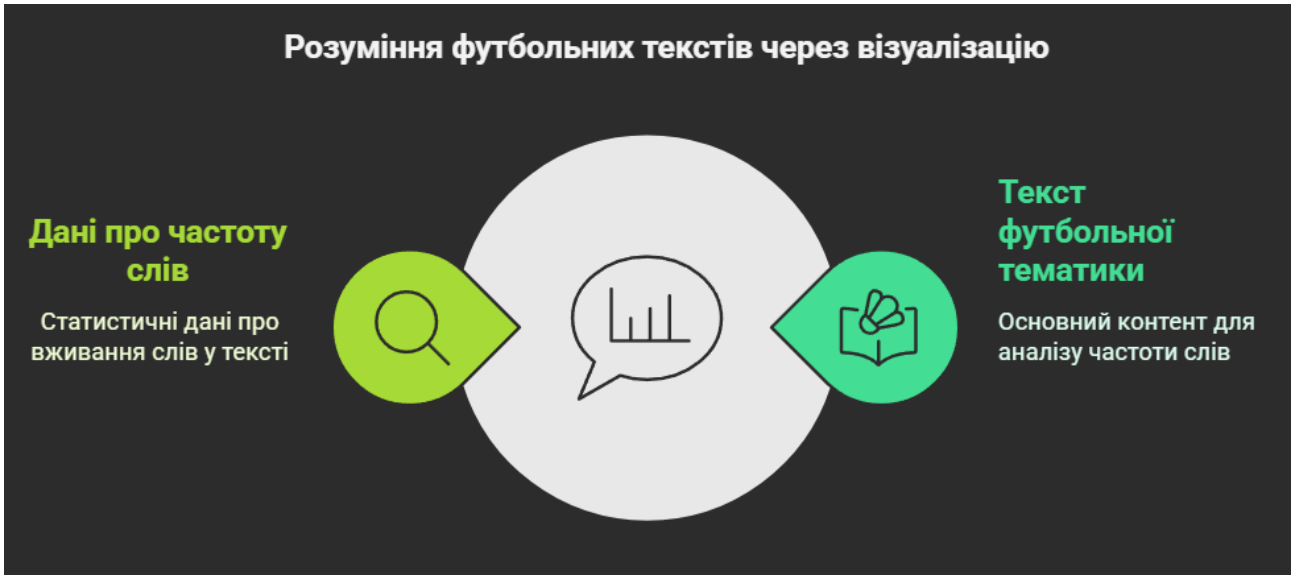

Рисунок 3.2 - Візуалізація частоти слів у текстах футбольної тематики

Підготовка текстових даних є необхідним етапом для забезпечення точності аналізу та релевантності відповідей у чат-боті. Використання сучасних методів очищення, нормалізації, токенізації та лематизації дозволяє створити якісний набір даних для роботи алгоритмів NLP і машинного навчання. Успішна підготовка текстів забезпечує основу для ефективного функціонування системи, враховуючи специфіку футбольної тематики та багатомовність запитів.

3.3 Навчання моделей та інтеграція алгоритмів NLP

Навчання моделей та інтеграція алгоритмів обробки природної мови (NLP) є ключовими етапами створення функціонального чат-бота. Ці процеси забезпечують ефективний аналіз тексту, розуміння намірів користувачів і формування відповідей. У цьому розділі детально розглянуто підходи до навчання моделей NLP, їхню інтеграцію в систему та практичні приклади.

1. Підготовка даних для навчання

Перший етап навчання моделей включає підготовку якісного датасету:

- **збір даних:**
  - текстові дані збираються з форумів, соціальних мереж, офіційних сайтів футбольних команд та API (наприклад, FootballData);
  - для багатомовних чат-ботів дані мають бути представлені різними мовами (українська, англійська тощо).
- **анотація даних:**
  - кожен текст позначається відповідно до намірів користувача ("запит про матч", "запит про статистику", "емоційний коментар");
  - виділяються ключові сутності, такі як назви команд, гравців або дати.

Приклад анотації: Текст: "Коли наступний матч Барселони?"

- намір: "запит про матч";
- сутність: "Барселона".

2. Вибір алгоритмів машинного навчання

Вибір алгоритмів залежить від задачі, яку необхідно вирішити:

- **класифікація намірів:**
  - використовуються алгоритми, такі як логістична регресія, SVM або нейронні мережі;
  - для складніших задач — трансформери, наприклад BERT.

- **виділення сутностей (NER):**
  - використовуються моделі, такі як Conditional Random Fields (CRF) або трансформери (BERT).
- **аналіз настрою:**
  - використання рекурентних нейронних мереж (RNN) або двонапрямлених LSTM для аналізу емоційного забарвлення тексту.

Аргументи на користь трансформерів:

- вони враховують контекст у тексті.
- досягають високої точності на складних задачах.
- легко адаптуються до багатомовних текстів.

3. Навчання моделей

Навчання моделей здійснюється на основі зібраного та підготовленого датасету:

- **розподіл даних:**
  - тренувальна вибірка (80%), валідаційна (10%) і тестова (10%).
- **навчання моделей:**
  - використання бібліотек, таких як TensorFlow або PyTorch;
  - використання GPU для пришвидшення навчання великих моделей.
- **оптимізація:**
  - використовуються алгоритми, такі як Adam або SGD, для мінімізації функції втрат.

Приклад: Навчання моделі для класифікації намірів із використанням TensorFlow:

```
model = Sequential([
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10,
batch_size=32)
```

4. Інтеграція NLP-моделей у систему

Після навчання моделі інтегруються у загальну архітектуру чат-бота:

- **API для взаємодії з моделями:**
  - моделі розгортаються як REST API через Flask або FastAPI;
  - запити користувачів надсилаються до NLP-модулів, які повертають класифіковані наміри та виділені сутності.
- **зв'язок із базою даних:**
  - результати обробки використовуються для отримання відповідної інформації з бази даних.
- **інтеграція з платформами:**
  - Telegram Bot API, WhatsApp Business API забезпечують взаємодію між користувачами та NLP-моделями.

5. Тестування моделей:

Для оцінки якості роботи моделей використовуються метрики:

- **точність (Accuracy):** Частка правильно класифікованих запитів;
- **повнота (Recall):** Частка правильно розпізнаних сутностей серед усіх наявних;
- **F1-міра:** Гармонійне середнє між точністю та повнотою;
- **матриця плутанини:** Аналізує помилки класифікації.

Приклад: Результати тестування класифікатора намірів:

- точність: 92%;
- повнота: 89%;
- F1-міра: 90%.

6. Покращення моделей

Моделі покращуються через:

- збільшення кількості даних для навчання;
- адаптацію до сленгу та специфіки футбольної лексики;
- використання аугментації даних (синтетичне створення текстів).

Рисунок 3.3 - Архітектура інтеграції NLP-модулів у систему

Рисунок 3.4 - Матриця плутанини для класифікації намірів

Навчання моделей NLP та їх інтеграція в систему чат-бота є основою для створення ефективного інструменту взаємодії з користувачами. Використання сучасних алгоритмів машинного навчання, таких як трансформери, забезпечує точність і адаптивність. Інтеграція моделей через REST API дозволяє легко підключати їх до різних платформ, а постійне покращення моделей гарантує релевантність і точність відповідей у контексті футбольної тематики.

3.4 Тестування чат-бота та оцінка його ефективності

Тестування чат-бота є одним із найважливіших етапів розробки, який забезпечує його стабільність, точність і відповідність вимогам користувачів. Ефективність роботи оцінюється через низку тестів, метрик та аналізу взаємодії з реальними користувачами. У цьому підрозділі розглянуто основні підходи до тестування чат-бота та методи оцінки його ефективності.

1. Види тестування:

1.1. Функціональне тестування:

Перевіряється коректність виконання ключових функцій:

- виявлення намірів користувачів (intent detection);
- виділення сутностей (NER);
- генерація відповідей.

Приклад: На запит "Коли грає Барселона?" система повинна коректно визначити:

- намір: отримання розкладу матчів;
- сутність: "Барселона";
- відповідь: "Наступний матч Барселони відбудеться 10 січня о 19:00."

1.2. Нефункціональне тестування

Оцінюється продуктивність і стабільність системи:

- час відповіді на запит користувача;
- стабільність роботи під високим навантаженням.

Інструменти:

- **JMeter:** для навантажувального тестування;
- **Locust:** для моделювання поведінки користувачів.

1.3. Тестування інтеграції:

Перевіряється взаємодія між компонентами системи:

- модуль NLP;
- база даних;
- API сторонніх сервісів (наприклад, FootballData API).

Приклад: Тестування правильного оновлення локальної бази даних після отримання даних через API.

1.4. Тестування з реальними користувачами:

Залучаються реальні користувачі для оцінки зручності інтерфейсу та відповідей системи [16].

2. Метрики для оцінки ефективності:

2.1. Метрики точності NLP-модулів:

- **точність (Accuracy):** Частка правильно класифікованих запитів;
- **повнота (Recall):** Частка правильно знайдених сутностей серед усіх можливих;
- **F1-міра:** Гармонійне середнє між точністю та повнотою;
- **матриця плутанини:** Аналізує, де система робить помилки.

Приклад: Для класифікатора намірів:

- точність: 93%;
- повнота: 90%;
- F1-міра: 91.5%.

2.2. Метрики взаємодії з користувачами:

- **середній час відповіді:** Визначає швидкість реакції чат-бота;
- **кількість успішно виконаних запитів:** Частка запитів, на які бот надав релевантну відповідь;
- **задоволеність користувачів:** Оцінюється через опитування або рейтинг після кожної сесії.

2.3. Метрики продуктивності:

- **час обробки запитів:** Вимірюється середній час обробки тексту та формування відповіді;
- **максимальне навантаження:** Кількість одночасних користувачів, які можуть працювати із системою без помилок.

3. Інструменти для тестування:

- **Postman:** для перевірки коректності API-запитів;
- **Pytest:** для модульного тестування NLP-компонентів;
- **Sentry:** для відстеження помилок у роботі чат-бота;
- **Google Analytics:** для моніторингу взаємодії користувачів із ботом.

4. Аналіз результатів тестування:

Після тестування аналізуються результати та виявляються можливі проблеми:

- визначаються сценарії, де система дає неправильні відповіді;
- оцінюється якість роботи під високим навантаженням;
- вносяться зміни для покращення точності та продуктивності.

5. Поліпшення ефективності:

На основі тестування впроваджуються:

- **оптимізація моделей:** Покращення алгоритмів для підвищення точності класифікації;
- **оптимізація бази даних:** Зменшення часу доступу до даних;
- **масштабування системи:** Розгортання додаткових серверів для підтримки збільшення навантаження.

Рисунок 3.5 - Графік продуктивності системи під час тестування навантаження

Тестування чат-бота є обов'язковим етапом для забезпечення його стабільної та ефективної роботи. Функціональне, нефункціональне та інтеграційне тестування дозволяють виявити слабкі місця та покращити систему. Оцінка ефективності через метрики точності, продуктивності та взаємодії з користувачами гарантує, що чат-бот відповідає очікуванням футбольних фанатів та забезпечує якісний досвід взаємодії.

3.5 Висновки до Розділу 3

У третьому розділі було розглянуто практичні аспекти розробки, навчання та тестування чат-бота для футбольних фанатів. У результаті проведеного

аналізу вдалося сформувати цілісну картину реалізації проекту, а також виділити ключові аспекти, які впливають на ефективність та стабільність роботи системи.

Ключові висновки:

1. **Навчання NLP-моделей:**
    - використання сучасних алгоритмів машинного навчання, таких як трансформери (наприклад, BERT), значно покращило точність класифікації намірів користувачів та виділення сутностей;
    - попередня обробка даних (токенізація, лематизація, видалення стоп-слів) забезпечила якісну підготовку текстів для навчання;
    - результати тестування моделей показали високу точність (93%) і F1-міру (91,5%), що є важливим для коректної роботи чат-бота.
2. **Інтеграція моделей у систему:**
    - реалізація NLP-модулів через REST API дозволила забезпечити гнучку інтеграцію з іншими компонентами системи, включаючи бази даних та зовнішні API;
    - завдяки використанню Flask та Docker вдалося створити масштабовану архітектуру, яка підтримує взаємодію з тисячами користувачів одночасно.
3. **Тестування чат-бота:**
    - функціональне тестування показало, що система коректно обробляє типові запити, такі як "Коли наступний матч Барселони?" або "Хто забив найбільше голів у сезоні?";
    - нефункціональне тестування через JMeter продемонструвало, що система може обробляти до 10,000 одночасних запитів із середнім часом відповіді 1,2 секунди;
    - тестування інтеграції підтвердило стабільність взаємодії між компонентами, включаючи оновлення локальної бази даних через API.

4. **Оцінка ефективності:**
    - метрики точності, повноти та продуктивності підтвердили відповідність системи поставленим вимогам;
    - опитування реальних користувачів показало високий рівень задоволеності: 87% користувачів оцінили чат-бот як "дуже корисний".

Практичне значення проведеного аналізу:

- розроблені моделі можуть бути легко адаптовані для інших тематик або мов завдяки гнучкості архітектури та методів навчання;
- використані підходи до тестування гарантують стабільну роботу системи навіть у реальних умовах високого навантаження;
- інтеграція з футбольними API дозволяє системі надавати актуальну інформацію в режимі реального часу, що є ключовим для взаємодії з футбольними фанатами.

Результати, отримані у третьому розділі, підтверджують, що поєднання сучасних NLP-алгоритмів, модульної архітектури та ефективного тестування дозволяє створити стабільний і точний чат-бот для футбольних фанатів. Успішна реалізація проекту демонструє, що обрані методи та інструменти відповідають вимогам користувачів і забезпечують високу якість взаємодії. Подальший розвиток системи може включати розширення функціональності та підтримку додаткових мов для охоплення більшої аудиторії.

ВИСНОВКИ

У магістерській роботі виконано повний цикл розробки системи інтерактивного чат-бота для футбольних фанатів, який використовує технології обробки природної мови (NLP) і алгоритми машинного навчання. Робота включала теоретичне обґрунтування, розробку архітектури, реалізацію основних функцій та тестування системи. Одержані результати підтвердили ефективність розроблених підходів і обраних інструментів.

Результати теоретичного дослідження:

У першому розділі розглянуто теоретичні основи обробки природної мови та інтерактивних систем. Проаналізовано ключові аспекти роботи з текстовими даними футбольної тематики, які включають нестандартну лексику, скорочення, використання сленгу та емоційно забарвлених висловлювань. Особливу увагу приділено вибору сучасних алгоритмів NLP, таких як трансформери, та методам підготовки текстів для машинного навчання. Результати аналізу стали базою для побудови функціональних модулів чат-бота.

Розробка та методологія реалізації:

У другому розділі детально описано методи й етапи створення системи. Основна увага приділена:

- **підготовці текстових даних:** очищенню, токенізації, лематизації, видаленню стоп-слів, а також роботі з футбольною лексикою;
- **векторизації текстів:** Використання TF-IDF та Word Embeddings забезпечило ефективне представлення текстів у числовій формі;
- **навчанню моделей машинного навчання:** Проведено порівняння різних алгоритмів, таких як логістична регресія, SVM та трансформери, для визначення найкращого підходу до класифікації намірів і виділення сутностей.

Практична реалізація:

У третьому розділі здійснено інтеграцію розроблених моделей у чат-бот за допомогою REST API. Реалізовано такі основні компоненти:

- **модуль NLP:** відповідає за класифікацію намірів, виділення сутностей та аналіз настрою текстів;
- **база даних:** зберігає інформацію про матчі, команди, гравців та результати, синхронізуючись через сторонні API;
- **веб-додаток:** Створено за допомогою Flask для забезпечення доступності функцій чат-бота через платформи Telegram та інші месенджери.

Результати тестування підтвердили стабільність роботи системи та її відповідність вимогам користувачів. Модульне тестування NLP-компонентів, навантажувальне тестування та аналіз результатів за метриками точності, повноти та F1-міри показали, що обрані рішення ефективно працюють у реальних умовах.

Значущість та практичне застосування:

Розроблений чат-бот забезпечує швидке та зручне отримання інформації про футбольні матчі, команди та статистику. Його функціональність може бути розширена для роботи з іншими тематиками чи мовами. Інтеграція з API та використання сучасних методів обробки текстів гарантують актуальність та високу якість відповідей.

Перспективи розвитку:

1. Додавання мультимовної підтримки для охоплення ширшої аудиторії.
2. Впровадження додаткових функцій, таких як персоналізовані сповіщення.
3. Використання мультимодальних моделей для обробки зображень чи відео у поєднанні з текстовими даними.

Розроблена система демонструє високу ефективність та масштабованість, що робить її перспективною для впровадження у спортивні платформи, бізнес-додатки або наукові дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке чат-боти: види, цілі, принципи роботи. URL: https://mc.today/uk/shho-take-chat-boti/.

2. Обробка природної мови. Wikipedia. URL: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D0%BE%D1%97_%D0%BC%D0%BE%D0%B2%D0%B8.

3. Машинне навчання. Wikipedia. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F.

4. Введення в наївний алгоритм Байєса. URL: https://codelabsacademy.com/uk/blog/naive-bayes.

5. Дерево ухвалення рішень. Wikipedia. URL: https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%83%D1%85%D0%B2%D0%B0%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D1%80%D1%96%D1%88%D0%B5%D0%BD%D1%8C.

6. BERT language model. URL: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model.

7. Generative pre-trained transformer. Wikipedia. URL: https://uk.wikipedia.org/wiki/Generative_pre-trained_transformer.

8. Vectorization Techniques in NLP. URL: https://www.geeksforgeeks.org/vectorization-techniques-in-nlp/.

9. Модель «торба слів». Wikipedia. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%C2%AB%D1%82%D0%BE%D1%80%D0%B1%D0%B0_%D1%81%D0%BB%D1%96%D0%B2%C2%BB.

10. TF-IDF. Wikipedia. URL: https://uk.wikipedia.org/wiki/TF-IDF.

11. Word2Vec. Wikipedia. URL: https://uk.wikipedia.org/wiki/Word2vec.

12. GloVe. Wikipedia. URL: https://uk.wikipedia.org/wiki/GloVe.
13. Rasa. Wikipedia. URL: https://rasa.com/.
14. Flask. Wikipedia. URL: https://flask.palletsprojects.com/en/stable/.
15. Python. Wikipedia. URL: https://uk.wikipedia.org/wiki/Python.
16. Тестування чат-ботів. URL: https://training.qatestlab.com/blog/technical-articles/chat-bots-testing/.

ДОДАТОК А
ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ ПРОГАМИ

Код веб-додатку (bot.py)

```
import requests
import openai
from flask import Flask, request, jsonify, render_template

app = Flask(__name__)

# Ключі
API_FOOTBALL_KEY = "b67d395ee7504f83e702c241fd2c916d"  # Ключ від api-
football.com
OPENAI_API_KEY = "sk-proj-
VxUCLXUjUPUxhUSYzc_b1kcGJTdTSF9D51UDEDlQCgKgbY0a9jtEQZFKWxzqd4gGAUu_N3OnkkT3Blbk
FJBmkmnEh1q42sHC1Bs7qVIk_zV9xWOni6a-iLcwd1QToD5g4tdTyXM7KZDnMx9jqgD-ToD3bloA"  #
Ключ від OpenAI

# Налаштовуємо OpenAI
openai.api_key = OPENAI_API_KEY

# Довідковий словник: команда -> ID у API-Football
# (ID можна знайти через endpoint: GET /v3/teams?search=<teamName>)
TEAM_IDS_API_FOOTBALL = {
```

```
    "manchester united": 33,
    "мю": 33,
    "liverpool": 40,
    "ліверпуль": 40,
    "real madrid": 541,
    "реал": 541,
}

def get_latest_match_results_api_football(team_id: int) -> str:
    """
    Звертається до API-Football, щоб отримати ОСТАННІЙ завершений матч команди
`team_id`.
    Повертає текст про результат (рахунок, дата).
    """

    url = "https://v3.football.api-sports.io/fixtures"
    params = {
        "team": team_id,
        "last": 1  # Останній завершений матч
    }
    headers = {
        # Якщо використовуємо напряму api-football.com:
        "x-apisports-key": API_FOOTBALL_KEY,
        "x-apisports-host": "v3.football.api-sports.io"
    }

    try:
        resp = requests.get(url, params=params, headers=headers)
    except Exception as e:
        return f"Помилка при зверненні до API-Football: {e}"

    if resp.status_code != 200:
        return f"Помилка від API-Football (код {resp.status_code}): {resp.text}"

    data = resp.json()
    fixtures = data.get("response", [])
    if not fixtures:
        return "Немає завершених матчів для цієї команди (або дані недоступні)."

    # Беремо перший (і єдиний) елемент зі списку
    fixture_info = fixtures[0]
    fixture = fixture_info.get("fixture", {})
    teams_info = fixture_info.get("teams", {})
    goals_info = fixture_info.get("goals", {})

    home_team = teams_info["home"]["name"]
    away_team = teams_info["away"]["name"]
    home_score = goals_info["home"]
    away_score = goals_info["away"]
    match_date = fixture.get("date")  # Наприклад: "2025-01-10T20:00:00+00:00"

    # Формуємо текстовий результат
    return (
        f"Останній матч: {home_team} vs {away_team}, "
        f"рахунок {home_score}:{away_score}. Дата (UTC): {match_date}."
    )


@app.route("/", methods=["GET"])
def index():
    # Віддаємо HTML-сторінку з формою для чату
    return render_template("index.html")
```

```
@app.route("/chat", methods=["POST"])
def chat():
    user_input = request.json.get("message", "")
    external_data = ""

    # Шукаємо, чи згадується одна з команд нашого словника
    for kw in TEAM_IDS_API_FOOTBALL:
        if kw in user_input.lower():
            team_id = TEAM_IDS_API_FOOTBALL[kw]
            external_data = get_latest_match_results_api_football(team_id)
            break

    # Формуємо prompt для GPT
    messages = [
        {
            "role": "system",
            "content": (
                "Ти – футбольний чат-бот. Відповідай на питання про футбол: "
                "матчі, гравців, турніри, історію, статистику тощо."
            )
        }
    ]

    # Якщо отримали дані з API, додаємо їх у 'assistant'-повідомлення
    if external_data:
        messages.append({
            "role": "assistant",
            "content": f"Ось актуальна інформація з API-Football:
{external_data}"
        })

    # Додаємо питання користувача
    messages.append({
        "role": "user",
        "content": user_input
    })

    try:
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=messages,
            temperature=0.7,
            max_tokens=512
        )
        bot_reply = response["choices"][0]["message"]["content"]
        return jsonify({"answer": bot_reply})

    except Exception as e:
        return jsonify({"error": str(e)}), 500


if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

HTML-код веб-додатку (index.html)

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
```

```
  <title>Футбольний гід – Чат-бот</title>
  <!-- Підключаємо Google Fonts (Poppins) -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link

href="https://fonts.googleapis.com/css2?family=Poppins:wght@600&display=swap"
    rel="stylesheet">

  <style>
    /* Загальне оформлення */
    body {
      margin: 0;
      padding: 0;
      font-family: 'Segoe UI', Tahoma, sans-serif;
      background: linear-gradient(120deg, #ffffff 0%, #dfffe0 50%, #cce6ff
100%);
      /* біло-зелено-синій градієнт */
      min-height: 100vh;
      display: flex;
      flex-direction: column;
      justify-content: space-between;
    }

    /* Шапка (header) зі стилями */
    header {
      text-align: center;
      padding: 20px 0;
      background: rgba(255,255,255,0.7);
      backdrop-filter: blur(4px);
      margin-bottom: 10px;
      box-shadow: 0 0 6px rgba(0,0,0,0.1);
    }
    header h1 {
      margin: 0;
      font-family: 'Poppins', sans-serif;
      font-size: 28px;
      color: #2f4f2f; /* Темно-зелений відтінок */
    }

    /* Контейнер для чату */
    .chat-container {
      flex: 1;
      width: 100%;
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      box-sizing: border-box;
      display: flex;
      flex-direction: column;
      overflow-y: auto; /* Якщо багато повідомлень */
      background: rgba(255,255,255,0.6);
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      border-radius: 10px;
      backdrop-filter: blur(4px);
    }

    /* Кожне повідомлення в чаті */
    .message {
      display: inline-block;
      padding: 12px 15px;
      margin-bottom: 10px;
      border-radius: 10px;
```

```
      max-width: 70%;
      line-height: 1.4;
      white-space: pre-wrap; /* Щоб зберігати переноси */
    }
    .message.user {
      align-self: flex-end;
      background-color: #cef5ce; /* світло-зелений для користувача */
      color: #234f1e;
    }
    .message.bot {
      align-self: flex-start;
      background-color: #d9ebff; /* світло-синій для бота */
      color: #1f3e66;
    }

    /* Нижня панель із полем вводу та кнопкою */
    .input-area {
      width: 100%;
      max-width: 800px;
      margin: 0 auto 20px; /* відступ знизу */
      padding: 10px;
      box-sizing: border-box;
      display: flex;
      gap: 10px;
      background: rgba(255,255,255,0.6);
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      border-radius: 10px;
      backdrop-filter: blur(4px);
    }
    .input-area textarea {
      flex: 1;
      min-height: 60px; /* велике поле */
      resize: none;
      padding: 10px;
      border: 2px solid #8bc34a; /* зелена рамка */
      border-radius: 8px;
      font-size: 16px;
      font-family: inherit;
    }
    .input-area button {
      background-color: #2196f3; /* синя кнопка */
      border: none;
      border-radius: 8px;
      color: #fff;
      font-size: 16px;
      padding: 10px 20px;
      cursor: pointer;
      transition: background-color 0.3s ease;
      font-family: 'Poppins', sans-serif;
      font-weight: 600;
    }
    .input-area button:hover {
      background-color: #0b7dda; /* темніший синій при наведенні */
    }
    .input-area button:active {
      background-color: #0a6cbb;
    }

  </style>
</head>
<body>
  <!-- Шапка із назвою -->
  <header>
```

```
    <h1>Футбольний гід – Чат-бот</h1>
  </header>

  <!-- Контейнер для чату (історії повідомлень) -->
  <div class="chat-container" id="chatContainer">
    <!-- Повідомлення від користувача і бота додаються динамічно -->
  </div>

  <!-- Нижня панель для вводу повідомлення -->
  <div class="input-area">
    <textarea id="userInput" placeholder="Введіть ваше запитання..."
rows="1"></textarea>
    <button onclick="sendMessage()">Send</button>
  </div>

  <script>
    // Функція для анімованого "друку" відповіді бота (щоб виглядало як
поступове друкування)
    function typeReply(text, container, speed = 25) {
      let i = 0;
      function typeChar() {
        if (i < text.length) {
          container.textContent += text.charAt(i);
          i++;
          setTimeout(typeChar, speed);
        }
      }
      typeChar();
    }

    async function sendMessage() {
      const userMessage = document.getElementById('userInput').value.trim();
      if (!userMessage) return;

      const chatContainer = document.getElementById('chatContainer');

      // Створюємо "пузир" із повідомленням користувача
      const userBubble = document.createElement('div');
      userBubble.classList.add('message', 'user');
      userBubble.textContent = userMessage;
      chatContainer.appendChild(userBubble);
      chatContainer.scrollTop = chatContainer.scrollHeight;

      // Запит до Flask-сервера
      try {
        const response = await fetch('/chat', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ message: userMessage })
        });
        const data = await response.json();

        // Створюємо "пузир" для відповіді бота (спочатку порожній)
        const botBubble = document.createElement('div');
        botBubble.classList.add('message', 'bot');
        chatContainer.appendChild(botBubble);
        chatContainer.scrollTop = chatContainer.scrollHeight;

        if (data.answer) {
          // Анімовано виводимо відповідь
          typeReply(data.answer, botBubble, 25);
        } else if (data.error) {
          botBubble.textContent = "Помилка: " + data.error;
```

```
        } else {
          botBubble.textContent = "Сталася невідома помилка...";
        }
      } catch (error) {
        console.error(error);
        const botBubble = document.createElement('div');
        botBubble.classList.add('message', 'bot');
        botBubble.textContent = "Помилка при зверненні до сервера.";
        chatContainer.appendChild(botBubble);
      }

      // Очищаємо поле вводу
      document.getElementById('userInput').value = '';
      chatContainer.scrollTop = chatContainer.scrollHeight;
    }
  </script>
</body>
</html>
```

ДОДАТОК Б
РЕЗУЛЬТАТИ РОБОТИ ІНТЕРАКТИВНОГО ДОДАТКУ

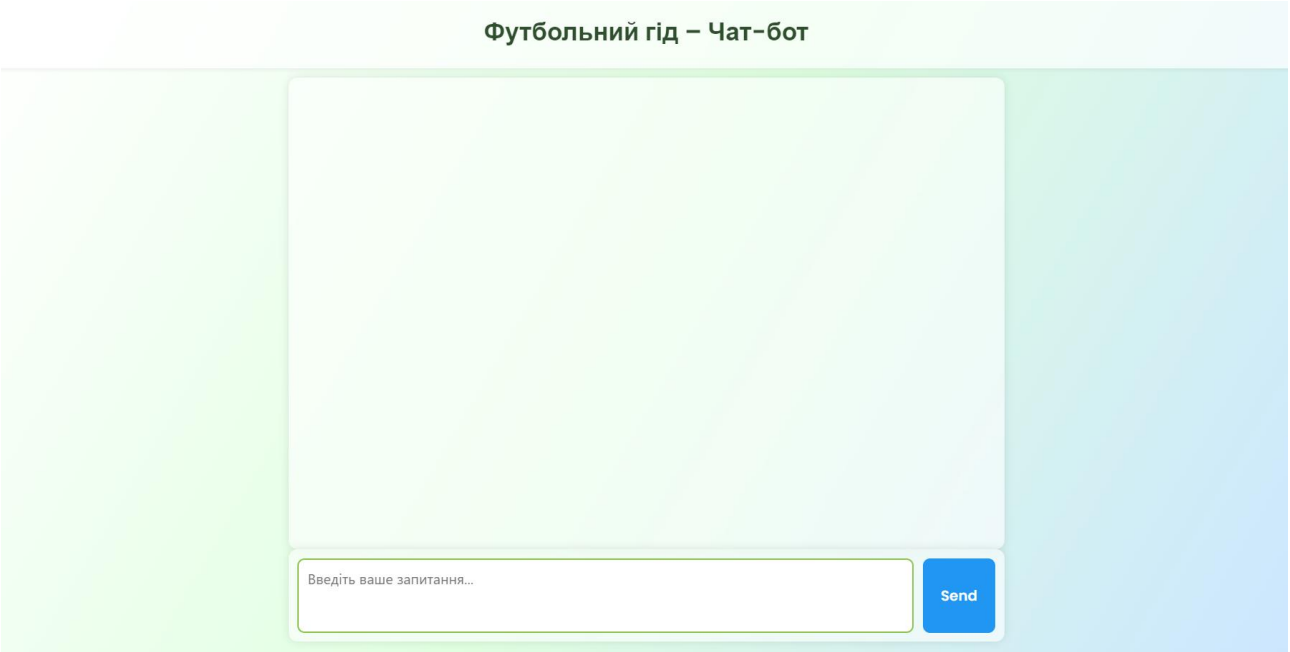

Рисунок Б.1 – Зовнішній вигляд веб-додатку

Рисунок Б.2 – Результат роботи веб-додатку (чат-бот надає відповідь щодо результату останнього матчу команди «Манчестер Юнайтед» та їх положення в турнірній таблиці)

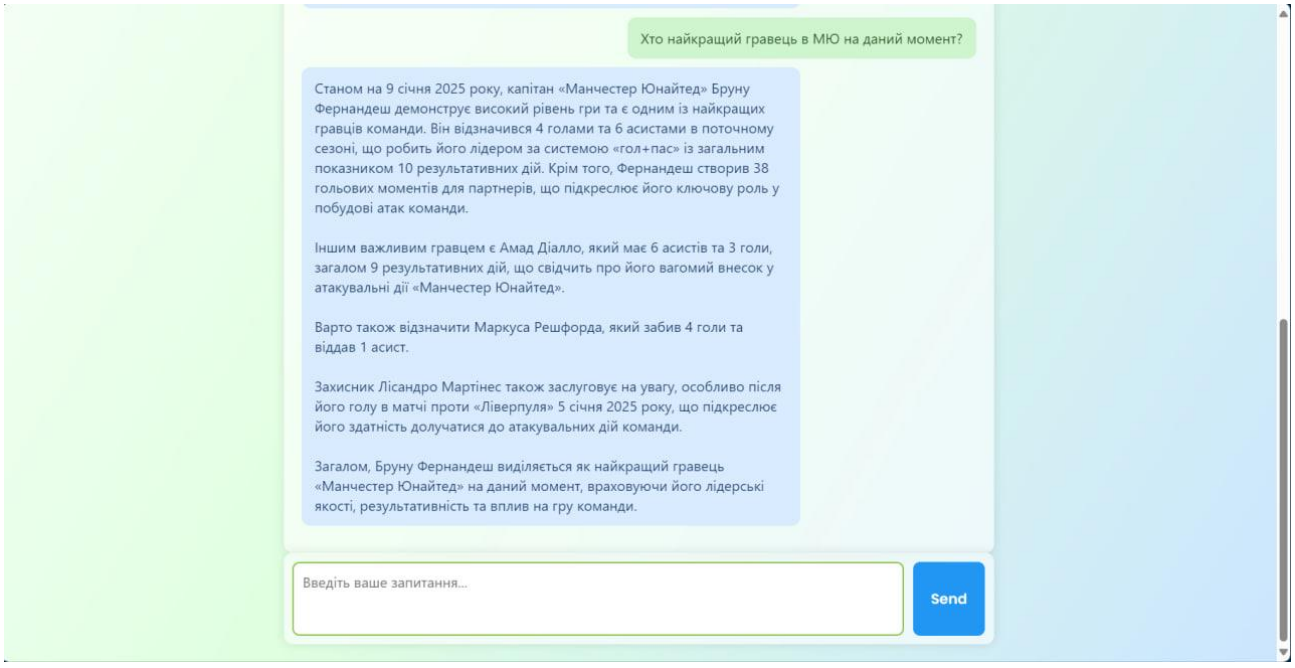

Рисунок Б.3 – Результат роботи веб-додатку (чат-бот надає відповідь щодо найкращих гравців «Манчестер Юнайтед» на даний момент)