**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«AI-помічник для автоматизованої технічної підтримки на основі нейронних мереж»**

на здобуття освітнього ступеня бакалавра
зі спеціальності *122 Комп'ютерні науки*
*(код, найменування спеціальності)*

освітньо-професійної програми *штучний інтелект*
*(назва програми)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*
_____ Ілля РУСИН
*(підпис)*

Виконав: здобувач вищої освіти групи ШІД-41
Ілля РУСИН
*(прізвище, ім'я)*

Керівник Андрій ВІКАРЧУК
*(науковий ступінь, вчене звання, прізвище, ім'я)*

Рецензент _____
*(науковий ступінь, вчене звання, прізвище, ім'я)*

Київ 2025

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри ШІ
_____ Ольга ЗІНЧЕНКО
"___" _____ 2025 року

# З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

Русин Ілля Костянтинович
*(прізвище, ім'я)*

1. Тема кваліфікаційної роботи: «AI-помічник для автоматизованої технічної підтримки на основі нейронних мереж»
керівник кваліфікаційної роботи Вікарчук Андрій , асистент
*(прізвище, ім'я, науковий ступінь, вчене звання)*
затверджені наказом Державного університету інформаційно-комунікаційних технологій від 24» лютого 2025 р. № 56

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 23.05.2025 р.

3. Вихідні дані до кваліфікаційної роботи
приклади реалізації AI-помічника у телеграмі;
наукові праці та публікації з тематики технічної підтримки, обробки природної мови.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи автоматизованої технічної підтримки
2. Розробка моделі AI-помічника
3. Реалізація та експерементальна оцінка моделі
4. Перелік графічного матеріалу
Презентація PowerPoint.

5. Дата видачі завдання «25» лютого 2025 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Визначення типів небажаних повідомлень у соціальних мережах | 15.03.2025 р. | Виконано |
| 2. | Аналіз наукової та технічної літератури з машинного навчання та NLP | 31.03.2025 р. | Виконано |
| 3. | Вибір алгоритмів та інструментів для побудови моделі класифікації | 10.04.2025 р. | Виконано |
| 4. | Формування датасету та підготовка навчальної вибірки | 25.04.2025 р. | Виконано |
| 5. | Розробка та тренування моделі виявлення небажаних повідомлень | 10.05.2025 р. | Виконано |
| 6. | Тестування системи та аналіз результатів | 25.05.2025 р. | Виконано |
| 7. | Підготовка звіту та доповіді до захисту | 31.05.2025 р. | Виконано |

Здобувач вищої освіти _____ Ілля РУСИН
*(підпис)* *(ім'я, прізвище)*

Керівник кваліфікаційної роботи _____ Андрій ВІКАРЧУК
*(підпис)* *(ім'я, прізвище)*

# РЕФЕРАТ

Текстова частина бакалаврської роботи: 78 сторінки, 30 рисунків, 20 джерел.

*Об'єкт дослідження* – процеси є процеси автоматизованої технічної підтримки клієнтів у великих інтернет-магазинах Rozetka, що включають обробку текстових запитів, надання інформації про товари, статуси замовлень та маршрутизацію складних питань до операторів.

*Предмет дослідження* – алгоритми обробки природної мови (NLP) на основі нейронних мереж, побудова моделей для розпізнавання намірів користувачів, а також інтеграція AI-помічника в платформи, такі як Telegram, для забезпечення ефективної технічної підтримки.

*Мета роботи* – є розробка та реалізація ефективного AI-помічника для автоматизованої технічної підтримки на основі нейронних мереж, а також проведення оцінки його ефективності шляхом експериментального тестування.

*Методи дослідження*: аналіз підходів до обробки природної мови — вивчення нейронних мереж, трансформерних архітектур та їхнього застосування в технічній підтримці; формування навчального датасету — обробка текстових запитів клієнтів, розмітка намірів; побудова моделі AI-помічника — проєктування та тренування системи з використанням Python та моделі від Together AI; тестування моделі — оцінка точності розпізнавання намірів, часу обробки запитів, рівня задоволеності користувачів, а також аналіз результатів на практичних прикладах.

*Галузь використання:* технічна підтримки, оптимізація клієнтського обслуговування та підвищення ефективності бізнес-процесів у сфері електронної комерції.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, АВТОМАТИЗАЦІЯ ТЕХНІЧНОЇ ПІДТРИМКИ, НЕЙРОННІ МЕРЕЖІ, NLP, КЛАСИФІКАЦІЯ

ЗАПИТІВ, AI-ПОМІЧНИК, ЕЛЕКТРОННА КОМЕРЦІЯ, ТЕХНІЧНА ПІДТРИМКА.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP - natural language processing (обробка природних мов)

ML - machine learning (машинне навчання)

AI - artificial intelligence (штучний інтелект)

RNN - recurrent neural network (рекурентні нейронні мережі)

CNN - convolutional neural network (згорткові нейронні мережі)

DL - deep learning (глибоке навчання)

НМ - нейронні мережі

Мб - мегабайт

CBOW - неперервна мультимножина зі словами

FNN - нейронна мережа прямого поширення

ANN - Artificial neural networks

LM - Language models (мовні моделі)

BPTT - back-propagation through time

**ЗМІСТ**

# ВСТУП

У сучасному, динамічному світі, що стрімко розвивається під впливом технологічних інновацій, автоматизована технічна підтримка набуває першочергового значення. Це особливо помітно на прикладі розширення діяльності великих інтернет-магазинів, таких як Rozetka, де щоденно зростає кількість звернень клієнтів. Активне впровадження штучного інтелекту (ШІ), зокрема передових технологій обробки природної мови (NLP), машинного навчання (ML) та глибинного навчання (DL), відкриває абсолютно нові горизонти. Ці технології дозволяють суттєво підвищити ефективність клієнтського обслуговування, значно скоротити операційні витрати, пов'язані з підтримкою, та забезпечити унікальний, персоналізований підхід до кожного користувача.

В умовах жорсткої конкуренції на ринку електронної комерції та постійного зростання обсягу клієнтських запитів, компанії змушені розробляти інноваційні інструменти. Ці інструменти мають бути здатними оперативно реагувати на звернення, адаптуватися до лінгвістичних та культурних особливостей різноманітної аудиторії та постійно оптимізувати процеси взаємодії з клієнтами. Саме тому створення ефективного AI-помічника для автоматизованої технічної підтримки є надзвичайно актуальним. Такий помічник повинен відповідати найвищим сучасним вимогам до якості обслуговування та забезпечувати конкурентні переваги для Rozetka, зміцнюючи її позиції на ринку.

Основною метою цієї роботи є всебічне дослідження, розробка, програмна реалізація та експериментальна оцінка AI-помічника. Цей помічник покликаний автоматизувати процеси технічної підтримки саме в інтернет-магазині Rozetka. Для досягнення цієї амбітної мети, було поставлено низку ключових завдань. Передусім, необхідно було провести глибокий теоретичний аналіз основних принципів та технологій автоматизованої технічної підтримки, включаючи детальний розгляд NLP, ML та DL. Далі, було потрібно розробити архітектуру

системи, яка б передбачала безперебійну інтеграцію із зовнішніми API та популярними платформами месенджерів.

Важливим кроком стала програмна реалізація чат-бота, використовуючи потужну модель meta-llama/Llama-3.3-70B-Instruct-Turbo-Free від Together AI, та його інтеграція у месенджер Telegram. Наступним етапом було експериментальне тестування системи. Це дозволило комплексно оцінити її функціональність, швидкість обробки запитів та, що найважливіше, рівень задоволеності користувачів. На основі отриманих результатів було запропоновано напрями подальшого вдосконалення системи, що є критично важливим для її довгострокової ефективності.

Об'єктом дослідження виступають існуючі процеси технічної підтримки в межах діяльності інтернет-магазину Rozetka. Предметом ж є безпосередньо розроблений AI-помічник, що реалізує автоматизовані функції підтримки, враховуючи як багатомовність, так і контекстуальні особливості діалогів з клієнтами.

Методологічна основа роботи ґрунтується на сучасних методах наукового аналізу, моделювання систем, програмування та експериментальної оцінки. Теоретична база побудована на ретельному вивченні наукової літератури, технічної документації та передових світових практик у сфері ШІ та NLP. Практична реалізація включає використання мови програмування Python, популярних бібліотек python-telegram-bot та together, а також реляційної бази даних SQLite для забезпечення надійної функціональності системи. Експериментальна оцінка передбачає всебічне тестування системи в реальних умовах з залученням симульованих користувачів. Це дозволяє аналізувати ключові метрики, такі як час обробки запитів, точність розпізнавання намірів користувачів та, звичайно, рівень задоволеності клієнтів.

Практична цінність цієї роботи полягає у створенні інструменту, що здатен ефективно автоматизувати типові сценарії технічної підтримки. До них належить надання інформації про товари, перевірка статусу замовлень, роз'яснення методів оплати тощо. Це значно сприятиме зменшенню навантаження на операторів

контакт-центру, дозволяючи їм зосередитись на складніших та нестандартних запитах.

Реалізована система має колосальний потенціал для підвищення рівня задоволеності клієнтів, що є абсолютно критичним фактором для успіху Rozetka як на українському, так і на міжнародних ринках електронної комерції. Крім того, результати цього дослідження можуть слугувати міцною основою для подальших розробок у сфері автоматизації клієнтського обслуговування. Це може включати, наприклад, інтеграцію з внутрішніми CRM-системами, обробку мультимодальних даних (таких як зображення чи голосові повідомлення) та розширення функціоналу для інших платформ.

Таким чином, ця робота не лише сприяє поглибленому теоретичному осмисленню проблематики, а й забезпечує практичне впровадження інноваційних рішень у реальне бізнес-середовище, що є її найвищою цінністю.

# 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗОВАНОЇ ТЕХНІЧНОЇ ПІДТРИМКИ

## 1.1 Принципи обробки природної мови в задачах технічної підтримки

Обробка природної мови (NLP) відіграє центральну роль в автоматизації технічної підтримки, тісно переплітаючись з іншими ключовими напрямами, такими як штучний інтелект (ШІ), машинне навчання (ML) та глибинне навчання (DL). У цьому розділі ми розглянемо принципи та методи, що забезпечують ефективну обробку як текстових, так і голосових запитів у сфері технічної підтримки, підкреслюючи їхню гармонійну інтеграцію.[1]

Штучний інтелект охоплює технології, що дозволяють комп'ютерним системам імітувати інтелектуальну поведінку людини. Це широке поле включає в себе машинне навчання, логічні міркування, планування та, звичайно, NLP. Наприклад, функція міркування дозволяє системам аналізувати вхідні запити та формувати гіпотези щодо можливих рішень, тоді як планування дає їм можливість автономно генерувати відповіді або ефективно скеровувати запити до відповідних фахівців.

У контексті технічної підтримки, NLP сфокусована на розумінні та обробці людської мови для автоматизації взаємодії з користувачами. Її застосування є багатогранним і охоплює такі важливі завдання, як класифікація запитів (наприклад, розрізнення технічних несправностей від консультацій), автоматичне створення відповідей, розпізнавання голосу в сучасних голосових помічниках та пошук інформації у величезних базах знань.[2]

NLP є критично важливою для систем технічної підтримки, адже вона охоплює всі аспекти взаємодії між комп'ютером і людиною через письмову чи усну мову. Однак, розуміння запитів ускладнюється через природну неоднозначність мови, яка може проявлятися у сарказмі, іронії або специфічній термінології. Для точної обробки та інтерпретації мови необхідно враховувати всі її компоненти. Фонетика аналізує звукові характеристики голосу, що є надзвичайно важливим для ефективної роботи голосових інтерфейсів

підтримки, наприклад, при розпізнаванні команд. Фонологія вивчає звукову систему мови, допомагаючи системам розпізнавати фонеми для коректного перетворення мовлення в текст. У технічній підтримці ці аспекти допомагають системі коректно інтерпретувати голосові запити користувачів. Морфологія досліджує структуру слів, дозволяючи системі, наприклад, перетворювати "підключений" у базову форму "підключити" для ефективного пошуку відповідної інформації в базі знань. Синтаксис аналізує порядок слів у реченні, що є критично важливим для побудови граматично правильних та логічно послідовних відповідей. Семантика визначає значення запитів, дозволяючи системі точно розрізняти, наприклад, "проблему з паролем" від "проблеми з мережею". Нарешті, прагматикавраховує широкий контекст спілкування, дозволяючи системі інтерпретувати запит "Чому не працює?" як прохання про діагностику проблеми, а не як риторичне питання.

Машинне навчання в технічній підтримці базується на концепції навчання з досвіду. Його визначення можна сформулювати так: система покращує свої результати за певною метрикою ефективності (P) для конкретної задачі (T), накопичуючи досвід (E). Наприклад, система класифікації запитів (T) використовує історичні дані технічної підтримки (E) для підвищення точності (P) у визначенні типу проблеми.

Машинне навчання в технічній підтримці поділяється на кілька основних парадигм: контрольоване навчання(наприклад, класифікація запитів з мітками), неконтрольоване навчання (групування схожих запитів без попередніх міток), посилене навчання (оптимізація відповідей через взаємодію з користувачами) та глибоке навчання (DL).

Глибоке навчання, використовуючи багатошарові нейронні мережі, дозволяє обробляти надзвичайно складні запити, наприклад, за допомогою потужних архітектур, таких як трансформери (BERT, GPT), для глибокого аналізу тексту чи генерації природних відповідей. У технічній підтримці DL ефективно вирішує задачі категоризації складних запитів та створення природних і зв'язних діалогів. Хоча DL є потужним інструментом, важливо зазначити, що простіші

моделі, як, наприклад, одношарові нейронні мережі, також можуть бути цілком застосовними для менш комплексних завдань, пропонуючи ефективні рішення для рутинних операцій.

Методи ШІ, ML, NLP та DL у технічній підтримці не існують ізольовано, а навпаки, взаємодіють у тісній синергії для створення по-справжньому ефективних систем. Наприклад, ML-алгоритми можуть класифікувати вхідні запити, NLP забезпечує глибоке розуміння тексту, а DL підвищує загальну точність завдяки здатності до глибокого аналізу даних і виявлення складних закономірностей. Ці фундаментальні принципи лежать в основі розробки сучасних AI-помічників, здатних автоматизувати обробку величезного потоку запитів, значно зменшуючи навантаження на операторів контакт-центру та, зрештою, підвищуючи якість обслуговування клієнтів.[3]

Токенізація є фундаментальним етапом обробки природної мови (NLP) у задачах технічної підтримки. Вона дозволяє нейронним мережам, часто реалізованим за допомогою таких фреймворків, як TensorFlow, ефективно розуміти зміст текстових запитів. Цей процес передбачає розбиття вхідних текстових даних на менші, керовані одиниці, так звані токени. Токени можуть бути окремими словами, частинами слів або навіть окремими символами, залежно від обраного методу токенізації. У технічній підтримці, де запити часто містять специфічну термінологію, різноманітні абревіатури чи неформальні вирази, токенізація відіграє вирішальну роль. Вона трансформує неструктурований текст у чіткий, структурований формат, який є придатним для подальшого аналізу та обробки нейронними мережами, що є першим кроком до розпізнавання намірів користувача та формування адекватної відповіді.

Однією з головних переваг токенізації є її здатність точно фіксувати семантичні зв'язки між словами, що критично важливо для розуміння людської мови комп'ютерними системами. Оскільки нейронні мережі оперують числовими даними, токенізація дозволяє представити кожен токен унікальним числовим ідентифікатором або вектором. Наприклад, запит "Проблема з підключенням Wi-Fi" може бути перетворений у послідовність

токенів ["проблема", "з", "підключенням", "Wi-Fi"], де кожному слову відповідає числове значення, наприклад, [10, 5, 22, 15]. Аналізуючи подібні числові представлення у великих наборах даних технічної підтримки, нейронна мережа здатна виявляти приховані закономірності. Це дозволяє їй асоціювати "Wi-Fi" з мережевими проблемами, що значно покращує точність класифікації запитів.

Токенізація також ефективно вирішує проблему слів поза словником (OOV), що особливо актуально в технічній підтримці, де постійно з'являються нові продукти, оновлення чи терміни. Наприклад, якщо користувач згадує невідоме слово, таке як назва нової моделі роутера, модель може обробити його токенізовану форму, розбивши на підслова, і зрозуміти його значення на основі контексту (наприклад, "не працює" чи "немає сигналу"). Цей підхід дозволяє AI-помічнику залишатися гнучким і адаптивним у реальних сценаріях, де запити часто містять унікальні або рідкісні терміни, забезпечуючи безперебійне обслуговування навіть у непередбачуваних ситуаціях.

Ще однією важливою функцією токенізації є її здатність обробляти запити змінної довжини, що є типовою характеристикою повідомлень у технічній підтримці. Запити можуть бути як надзвичайно короткими ("Не працює"), так і розгорнутими описами проблеми з безліччю деталей. Токенізація уніфікує ці дані, перетворюючи їх у послідовності токенів фіксованої довжини шляхом додавання заповнювачів (padding) або обрізання. Це забезпечує ефективну обробку нейронними мережами, які потребують однакової довжини вхідних даних для паралельних обчислень, що критично важливо для швидкої реакції в реальному часі та забезпечення безперебійної роботи системи.

Крім того, токенізація значно знижує обчислювальну складність, що є важливим фактором для ефективного масштабування систем технічної підтримки. Розбиваючи текст на токени, можна суттєво зменшити розмір словника, обмежуючи кількість унікальних елементів, які повинна обробляти модель. Наприклад, використання токенізації на основі підслів (як у Byte-Pair Encoding) дозволяє представляти рідкісні терміни через комбінації частих підслів. Це

ефективно зменшує розмірність даних і полегшує навчання складних моделей на великих наборах даних, роблячи процес обробки більш швидким та економічним.

Новим та надзвичайно перспективним підходом у технічній підтримці є контекстно-залежна токенізація, яка враховує специфіку домену. У запитах часто зустрічаються технічні терміни ("DHCP", "перепрошивка"), які стандартні токенізатори можуть неправильно розбити. Спеціалізовані токенізатори, навчені на великих корпусах технічних текстів, зберігають такі терміни як єдині токени, значно підвищуючи точність семантичного аналізу та уникнення помилок. Крім того, токенізація з урахуванням емоційного контексту дозволяє виявляти тональність запиту (наприклад, роздратування, терміновість чи навіть сарказм). Це допомагає AI-помічнику адаптувати свою відповідь, роблячи її більш чутливою та ефективною, що, в свою чергу, підвищує загальний рівень задоволеності користувача.

Таким чином, токенізація є абсолютно незамінним етапом у задачах технічної підтримки. Вона дозволяє нейронним мережам, побудованим, наприклад, на TensorFlow, обробляти текстові запити шляхом їх перетворення в числові формати. Цей процес забезпечує глибоке розуміння семантичних зв'язків, ефективну обробку OOV-слів, уніфікацію даних змінної довжини та значне зниження обчислювальної складності. Сучасні методи, такі як контекстно-залежна та емоційно-орієнтована токенізація, додатково підвищують ефективність AI-помічників, роблячи їх здатними до точної та адаптивної обробки складних запитів у реальних умовах.[4]

Штучний інтелект (ШІ) та технології обробки природної мови (NLP) відкривають абсолютно нові горизонти для поглиблення автоматизації в контакт-центрах. Вони дозволяють надавати швидкі початкові відповіді клієнтам, вирішувати їхні запити через цифрові платформи та точно спрямовувати звернення до найбільш відповідних операторів у разі потреби. Ці технології сприяють поступовому, але впевненому переходу від часткової до більш комплексної автоматизації. Це є важливим кроком, враховуючи сучасний рівень автоматизації, що вже досягнутий у контакт-центрах завдяки системам

інтерактивного голосового меню (IVR, Interactive Voice Response) та автоматичного розподілу дзвінків (ACD, Automatic Call Distribution).

Однак, незважаючи на їхню корисність, традиційні IVR-системи мають суттєві недоліки, які негативно впливають на клієнтський досвід. Дослідження показують, що складні та багатоступеневі IVR-меню часто викликають у клієнтів роздратування через їхню заплутаність, утилітарний підхід до взаємодії та часто незручний інтерфейс (рис. 1.1). Це підкреслює нагальну потребу в більш інтуїтивних та інтелектуальних рішеннях, які може запропонувати сучасний ШІ.

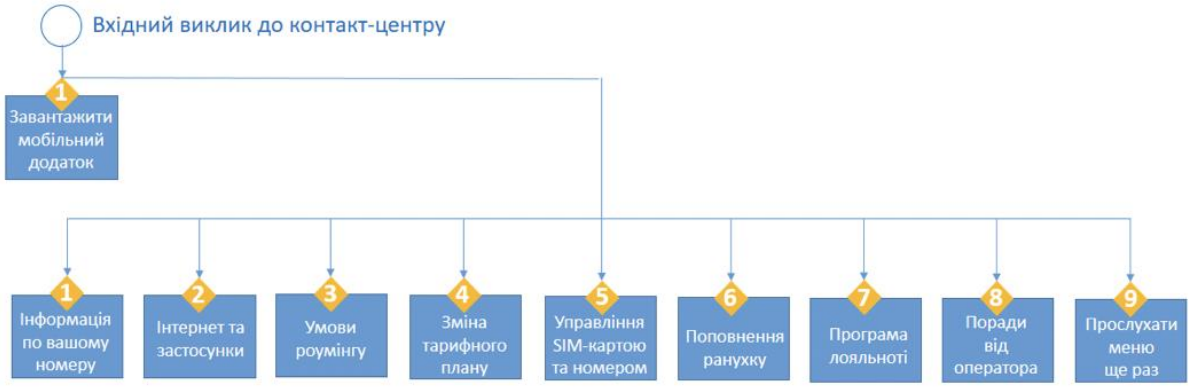


Рисунок 1.1 - IVR-меню телекомунікаційної компанії

Найбільш проблемним є те, що клієнти відчувають себе проігнорованими, оскільки опція зв’язку з оператором може бути недоступною або “захована” на останніх рівнях меню. Наприклад, типове IVR-меню телекомунікаційної компанії може включати вступну інформацію (наприклад, рекламу додатка) і кілька гілок опцій, але не надавати прямого доступу до оператора. У таких випадках відтворення меню може тривати до 60–70 секунд, що погіршує враження від сервісу. Це підкреслює потребу в нових рішеннях, які прискорюють взаємодію та дозволяють або швидко вирішити запит через автоматизацію, або оперативно з’єднати клієнта з оператором. [5]

Технології NLP, які можна розглядати як еволюцію IVR і ACD, пропонують більш інтуїтивний алгоритм взаємодії з контакт-центром (рис. 1.2):

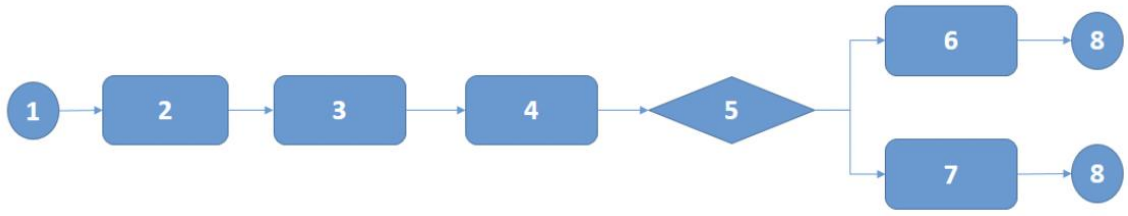


Рисунок 1.2 - Алгоритм взаємодії з контакт - центром

1. Надходження вхідного дзвінка.
2. Вітальне повідомлення від системи.
3. Отримання та аналіз запиту клієнта.
4. Обробка запиту з використанням NLP.
5. Маршрутизація дзвінка на основі результатів аналізу.
6. Надання автоматизованої консультації голосовим ботом.
7. З’єднання з оператором за необхідності.
8. Завершення обробки звернення.

Такий підхід значно перевершує традиційні IVR і ACD, оскільки NLP дозволяє системі розпізнавати природну мову, спрощуючи взаємодію клієнта з голосовим каналом. Завдяки аналізу мовлення та тексту в реальному часі, NLP забезпечує точну маршрутизацію, скорочує час очікування та зменшує ймовірність помилок у спрямуванні запитів. Це підвищує ефективність роботи контакт-центру та покращує клієнтський досвід, що є ключовим для підвищення рівня задоволеності.

Хоча часткова автоматизація (наприклад, оцінка якості звернень, аналіз потреб клієнтів або маршрутизація) уже застосовується, основний потенціал економії криється в повній автоматизації контакт-центрів за допомогою голосових і чат-ботів. Заробітна плата операторів становить приблизно 60% витрат контакт-центру, а підтримуючого персоналу — ще 20%. Заміна людських операторів ботами на основі NLP може суттєво знизити ці витрати, зберігаючи

або навіть підвищуючи якість обслуговування. Концепція “контакт-центру без операторів” стає реальною завдяки сучасним досягненням у ШІ.

Технології NLP мають величезний потенціал для трансформації контакт-центрів, забезпечуючи глибшу автоматизацію процесів, зниження витрат і покращення клієнтського досвіду. Від часткової автоматизації (маршрутизація, оцінка якості) до концепції “контакт-центру без операторів”, NLP-боти відкривають шлях до економії сотень мільярдів доларів щорічно. Однак успішне впровадження вимагає вирішення питань безпеки, якості даних і етичних викликів. Компанії, які активно інвестують у ці технології, отримають конкурентну перевагу, адаптуючись до зростаючого попиту на швидке, персоналізоване та ефективне обслуговування.[6]

## 1.2 Нейронні мережі для обробки текстових даних

Нейронні мережі є основою сучасних систем обробки природної мови (NLP), які широко застосовуються в задачах технічної підтримки для аналізу текстових запитів, їх класифікації, генерації відповідей і маршрутизації. У цьому розділі розглядаються основні типи нейронних мереж, їх архітектури та специфіка використання в обробці текстових даних, з акцентом на автоматизацію контакт-центрів. Особлива увага приділяється порівнянню ефективності різних моделей і новітнім підходам, які підвищують продуктивність AI-помічників.[7]

Згорткові нейронні мережі (CNN) - це клас глибинних штучних нейронних мереж прямого поширення, які формують основу комп’ютерного зору та обробки зображень. Вони були розроблені для більш ефективної обробки зображень. Це така нейронна мережа, в якій додаються додаткові шари у вигляді різних фільтрів та згорток. Вона складається з різних типів рівнів, які разом вивчають ієрархічні представлення вхідних даних. В CNN для вилучення ознак із зображень використовується операція згортки. Це особливість використовує спільні параметри шарів, де для обробки різних частин вхідного зображення використовуються однакові вагові коефіцієнти. Таким чином в нейронній мережі

зменшується кількість параметрів для навчання. Згорткові шари (див. рисунок вище) складаються з набору фільтрів, які можна розглядати як просто двовимірні матриці чисел. Операція згортки — це поелементне множення частини вхідного зображення на ядро фільтра, щоб отримати нову одну точку даних. Ми можемо використовувати вхідне зображення та фільтр для створення вихідного зображення шляхом згортання фільтра з вхідним зображенням. Це складається з:

1.Накладання фільтра поверх зображення в певному місці.

2.Виконання поелементного множення між значеннями у фільтрі та їхніми відповідними значеннями на зображенні.

3.Поелементне додавання результатів множення. Ця сума є вихідним значенням для пікселя призначення у вихідному зображення.

4.Повторення попередніх операцій для усіх пікселів.[8]

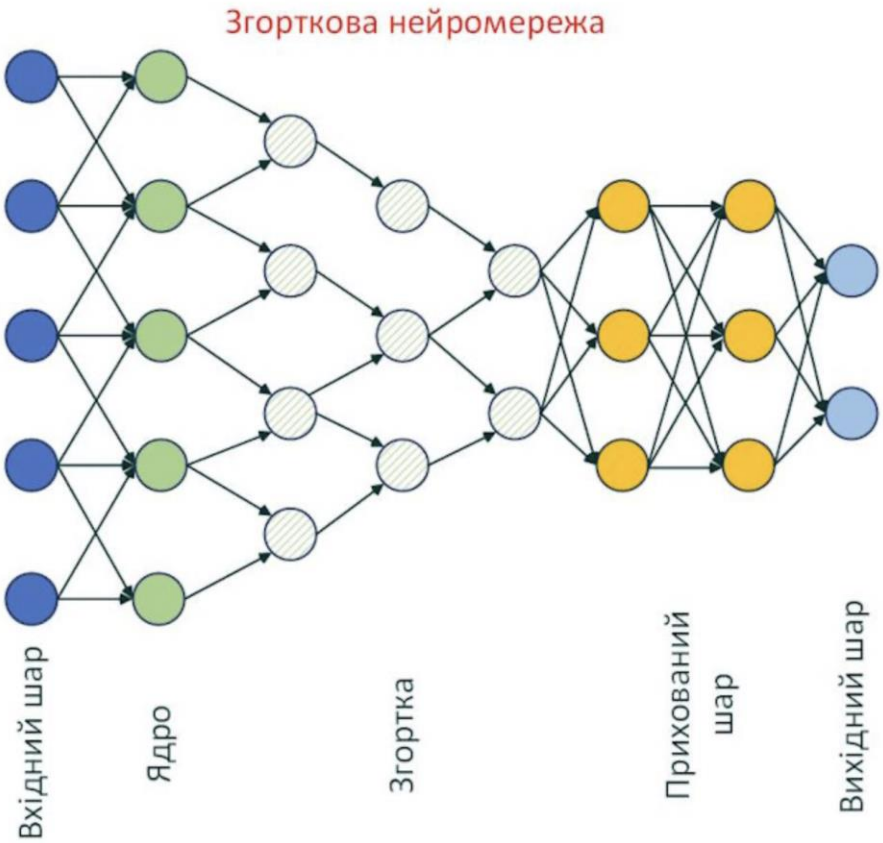


Рисунок 1.3 - Згорткова нейромережа

Рекурентні нейронні мережі (RNN). Повторювані нейронні мережі використовують загальний принцип прямої нейронної мережі та дозволяють їм

обробляти послідовні дані за допомогою надання моделі внутрішньої пам’яті. «Рекурентна» частина назви RNN походить від того факту, що вхід і вихід циклічні. Після створення вихідних даних мережі вони копіюються та повертаються в мережу як вхідні дані. При прийнятті рішення аналізуються не тільки поточний вхід і вихід, але також враховується попередній вхід. Іншими словами, якщо початковим входом для мережі є X, а виходом є H, і H, і X1 (наступний вхід у послідовності даних) подаються в мережу для наступного циклу навчання. Таким чином, контекст даних (попередні вхідні дані) зберігається під час навчання мережі. Результатом цієї архітектури є те, що RNN здатні обробляти послідовні дані. Однак RNN страждають від кількох проблем. RNN страждають від проблеми зі зникаючим градієнтом і вибуховим градієнтом. Довжина послідовностей, які може інтерпретувати RNN, досить обмежена, особливо в порівнянні з LSTM.

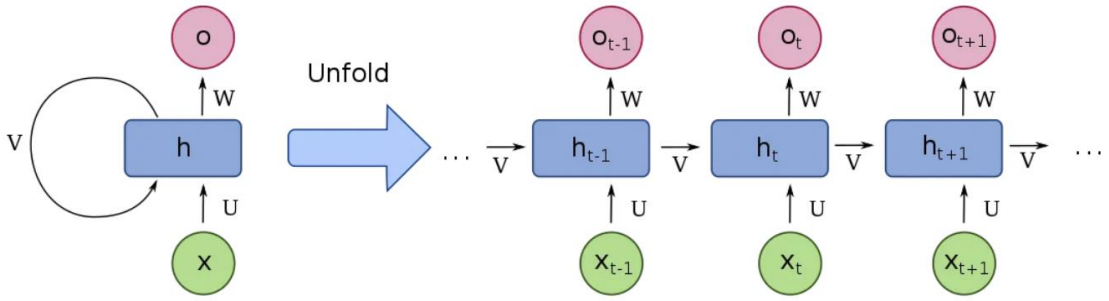


Рисунок 1.4 - Рекурентні нейронні мережі

Мережі довготривалої короткочасної пам’яті можна вважати розширенням RNN, ще раз застосовуючи концепцію збереження контексту вхідних даних. Однак LSTM були модифіковані кількома важливими способами, які дозволяють їм інтерпретувати минулі дані кращими методами.

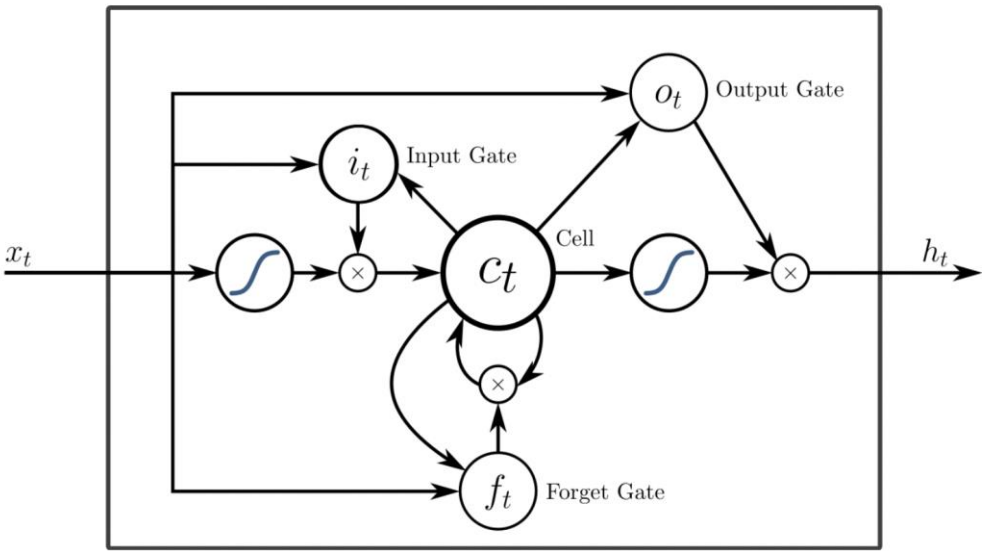


Рисунок 1.5 - Мережі довготривалої короткочасної пам’яті

Зміни, внесені до LSTM, стосуються проблеми зникнення градієнта та дозволяють LSTM розглядати набагато довші вхідні послідовності.

Моделі довготривалої короткочасної пам'яті (LSTM) складаються з трьох основних компонентів, які діють як "ворота", що контролюють потік інформації: вхідні ворота, вихідні ворота та ворота забуття. Подібно до рекурентних нейронних мереж (RNN), LSTM враховують вхідні дані з попереднього часового кроку під час модифікації внутрішньої пам'яті моделі та її вагових коефіцієнтів.

Вхідні ворота відіграють ключову роль у вирішенні того, які нові значення є важливими і мають бути інтегровані в пам'ять моделі. Цей процес здійснюється за допомогою сигмоїдної функції**,** яка фільтрує вхідні значення, пропускаючи лише ті, що мають значення (від 0 для відкидання до 1 для збереження). Паралельно, функція TanH визначає ступінь важливості цих вхідних значень для моделі, встановлюючи їх у діапазоні від -1 до 1. Після врахування поточних вхідних даних і стану пам'яті, вихідні ворота вирішують, яка інформація буде передана на наступний часовий крок. Тут значення аналізуються та їм призначається важливість у діапазоні від -1 до 1, що ефективно регулює дані перед їхнім використанням у наступних обчисленнях. Нарешті, ворота забуття

виконують критичну функцію: вони відповідають за видалення інформації, яку модель вважає непотрібною для прийняття поточного рішення. Ворота забуття також використовують сигмоїдну функцію, виводячи числа від 0 (повністю забути) до 1 (повністю зберегти). Загалом, нейронна мережа LSTM включає як спеціалізовані LSTM-шари, призначені для інтерпретації послідовних даних, так і щільно зв'язані шари. Коли дані проходять через LSTM-шари, вони згодом передаються до щільно з'єднаних шарів для подальшої обробки та формування кінцевого результату.[9]

Трансформери кардинально відрізняються від традиційних рекурентних нейронних мереж (RNN) своєю паралельною архітектурою, що забезпечує значно вищу ефективність. Основним будівельним блоком трансформерів є механізм наскрізного самовважання (self-attention), який дозволяє моделям захоплювати складні залежності між елементами послідовності, навіть якщо вони знаходяться далеко один від одного у тексті. Ця інноваційна архітектура революціонізувала сферу обробки природної мови (NLP), привносячи неперевершену ефективність та відкриваючи абсолютно нові можливості. Трансформери базуються на архітектурі нейронної мережі, спеціально розробленій для ефективної роботи з послідовностями даних, такими як текст.[20] Крім механізму самовважання, трансформери також використовують інші ключові компоненти для обробки даних:

- Енкодери положення**:** Ці компоненти вбудовують інформацію про відносне положення елементів у послідовності. Це життєво важливо, оскільки, на відміну від RNN, трансформери обробляють всю послідовність одночасно, а не послідовно, і тому їм потрібен окремий механізм для розуміння порядку слів.
- Мультиголовочні механізми самовважання**:** Цей механізм дозволяє моделі одночасно захоплювати різні типи та рівні взаємозалежностей між елементами в послідовності, розглядаючи їх з різних "перспектив" або "голосів".

- Позиційно-мультиплікативну увагу: Цей механізм допомагає посилювати або, навпаки, послаблювати увагу до певних елементів послідовності на основі їхнього положення, підвищуючи релевантність обробки та дозволяючи моделі краще фокусуватися на важливих частинах тексту.

Існує кілька відомих видів трансформерів, які стали основою для безлічі сучасних моделей у NLP:

- Transformer: Оригінальна модель, вперше запропонована компанією **Google**, що заклала фундамент для всієї подальшої архітектури.
- BERT (Bidirectional Encoder Representations from Transformers): Розроблений Google AI**,** цей двонаправлений трансформер-енкодер відмінно підходить для глибокого розуміння контексту слова в реченні, аналізуючи його як зліва направо, так і справа наліво.
- GPT (Generative Pre-trained Transformer): Розроблений компанією OpenAI**,** це генеративний попередньо навчений трансформер, який здобув широку популярність завдяки своїй здатності генерувати надзвичайно високоякісний, зв'язний та креативний текст.
- T5 (Text-to-Text Transformer): Також запропонований Google AI, цей уніфікований текстовий-текстовий трансформер перетворює практично всі задачі NLP на єдиний формат "текст на вхід - текст на вихід", що значно спрощує розробку та масштабування рішень.

Трансформери знайшли широке застосування в різних задачах NLP, включаючи:

- Класифікацію тексту: ефективне розподілення текстових документів за заданими категоріями.
- Екстракцію сутності: точне виявлення та вилучення важливих іменованих сутностей, таких як імена людей, назви організацій, дати та місця, з великих обсягів тексту.
- Генерацію мови: створення зв'язних, граматично правильних та змістовних текстових рядків на основі заданого вводу або певного контексту.

- Машинний переклад: високоякісний переклад текстових документів з однієї мови на іншу, зберігаючи при цьому семантичний зміст та стилістичні особливості оригіналу.
- Узагальнення текстів: автоматичне створення коротких і стислих версій довгих текстових документів, при цьому зберігаючи їхній основний зміст та ключові ідеї.

Ці передові нейронні мережі є фундаментальними для побудови інтелектуальних систем технічної підтримки, дозволяючи їм ефективно розуміти, обробляти та надавати точні й швидкі відповіді на запити користувачів, що значно покращує якість клієнтського обслуговування.[10]

### 1.3 Аналіз типових сценаріїв у технічній підтримці

Обробка запитів людей — основна складність під час створення чат-бота, і це можна зробити двома способами:

1) Задати строгий набір команд у форматі «запитання-відповідь», що суттєво лімітує систему. Це системи Rule-based, які можна порівняти з роботою веб-сторінок, чітко обмежених адресою.

2) Підключити машинне навчання, що допоможе моделі розуміти користувача за кількома ключовими словами. Це системи AI-based, в основі яких лежать моделі NLP (Natural language processing — системи обробки природної мови). Робота з NLP триває впродовж усього циклу розробки й поділяється на 4 фази: дискавері, тренування моделі, безпосередньо тестування та навчання після виходу в реліз.

Фаза 1. Дискавері

Тестування NLP починається на стадії збору вимог. Тут важливо визначити такі моменти:

- Мова. Впливає на вибір NLP-інструмента, кількість часу, який буде закладено в тест-план, залучення додаткових спеціалістів у команду.

- Feature scope. На час розробки та тестування на кількість фраз для перевірки впливає ще й те, який функціонал завдяки NLP-діалогам хоче покрити клієнт.
- Цільова аудиторія. Деталі про стать, вік, геолокацію, професію — показники, що допоможуть вибрати оптимальні висловлювання для тренування і тестування.

Фаза 2. Тренування моделі

Після затвердження вимог починається наповнення моделі фразами, виділення намірів користувача. Важливо працювати зі спеціалістами, для яких мова чат-бота рідна, вони допоможуть підібрати найоптимальніші фрази. А також — зі сторони клієнтів, які знають специфіку термінології свого домена і те, як користувачі спілкуються з їхнім бізнесом.

Фаза 3. Тестування

Починається найскладніше. Будьте готові до того, що ви стикнетеся з такими проблемами:

- Неможливо перевірити всі варіанти інпутів у систему (тобто всі комбінації слів, символів, речень, словосполук, які потенційно може ввести користувач). Під час тестування звичайних фіч використовується техніка поділу всієї множини вхідних даних на схожі підмножини, які мають подібні властивості і вибір по одному представнику з кожної. У випадку NLP є тисячі комбінацій слів на кожну з підмножин визначити одну або кілька неможливо.
- Помилки будуть завжди, адже складно визначити якість системи на різних етапах і виділити, що є багом. NLP-модель ніколи не відповідатиме повністю коректно. Обговоріть з клієнтом критерії якості цих моделей, щоб провести навчальну роботу з бізнесом і пояснити, як працюють такі технології і системи.

- Визначення критеріїв закінчення тестування. Це складно, адже можна тижнями перевіряти всі можливі висловлювання від користувача і не покрити навіть меншої частини вимог.

Для розв'язання цих проблем і ефективного тестування NLP ми виділили два методи:

- Визначення пріоритетів на основі аналізу ризиків. Це допомагає знайти належні приклади для тестування, зробити смоук- чи регрешн-тестування критичних ділянок і швидко дати фідбек стейкхолдерам про проблеми на проекті.
- Аналіз даних. Потрібно збирати статистику всього, що користувачі пишуть у бот, і виділити топ випадків, на які варто звернути увагу.

Фаза 4. Навчання на продакшені

Після релізу починається найцікавіша фаза тестування — збір статистики. Нам важливо розуміти:

- чи дійсно користувачі пишуть те, що ми очікували;
- які слова і фрази найпопулярніші;
- як система реагує на валідні та невалідні запити.

Після аналізу інформації потрібно:

- визначити, що варто додати, змінити або видалити в поточній моделі;
- перетренувати систему;
- зробити ще один раунд тестування, щоб зрозуміти, як поводитиметься нова модель.

Ця фаза може повторюватись 1-2 рази на місяць, залежно від нової функціональності, побажань клієнта та результатів попереднього аналізу.

QA-basic-чекліст

1. Помилки в словах

У доборі слів для тестування потрібно аналізувати ризики й вибирати найкритичніші варіанти. Наприклад, якщо функцією нашого асистента є доставка піци, обов'язково потрібно перевірити всі варіанти написання з помилками слів «order» та «pizza».

2. Слова-синоніми

Щоб дібрати оптимальну кількість слів, потрібно фокусуватися на найуживаніших синонімах. У цьому можуть допомогти словники та статистика юзер-інпутів. Синонімами також можуть бути цілі фрази.

3. Речення, які крім ключових фраз мають додаткові слова, що не належать до основного наміру користувача

У такому разі речення повинні вести на коректну відповідь на основі ключових фраз, а решту додаткових слів система повинна ігнорувати. Сюди часто потрапляють довгі й неконструктивні фрази, адже користувач не розуміє, що спілкується з неживим асистентом.

4. Речення, які крім ключових фраз мають смолтоки (smallTalks)

Цей кейс тестується окремо від попереднього, бо фрази смолтоку належать окремій групі слів — smallTalk. У такому разі QA має перевірити, чи система правильно розпізнає ключові слова й адекватно відповідає користувачам.

5. Різні наміри, але ті ж слова

Є ситуації, коли те ж саме слово є ключовим у різних намірах користувача. Вони можуть відрізнятися настільки, що вестимуть користувача на іншу відповідь. Така дія рівноцінна втраті користувача.[11]

Наприклад, користувач зробив покупку, не отримав доставку вчасно і написав у чат-бот: «I'd like to know where is my dress». Це trackOrder-намір. У такому випадку повести користувача на makeOrder-контент буде недоречно. Імовірно, що людина не зробить ще одну покупку, покине застосунок і звернеться у службу підтримки на веб-сайті.

6. Випадки, коли NLP мовчить

Попри те, що дружній бот повинен завжди вести діалог з користувачем і підтримувати його зацікавленість, є такі флоу, де NLP має мовчати. У цих точках

ми ставимо «заглушки», які його вмикають і вимикають. Потрібно враховувати непослідовність дій користувача і те, що він може вимкнути NLP назавжди. Тому в таких випадках ми користуємось тестом дизайну техніки переходу станів системи, щоб не пропустити жодної з послідовностей виконання дій.

7. Випадки, коли NLP допомагає користувачу не заблукати в структурі бота

Це актуально тоді, коли асистент має послідовну структуру (тобто після першого кроку юзер відразу переходить до другого). Тут необхідно врахувати, що користувач може випадково вийти з бота, не завершивши всі кроки. Ми пропонуємо додавати повторні запитання, щоб повернути його в попередню точку. Але щоб «перепитування» не дратувало користувача, варто дібрати кілька варіантів одного питання. Тут обов'язково передбачити вихід із циклу для користувачів, які мають інший запит — кнопки «back» чи «exit».

## 1.4 Сучасні системи автоматизованої технічної підтримки

Сучасні системи автоматизованої технічної підтримки активно використовують передові технології штучного інтелекту (ШІ) та обробки природної мови (NLP). Їхня головна мета — оптимізувати обробку клієнтських запитів, значно підвищити якість обслуговування та суттєво знизити операційні витрати. Цей розділ аналізує провідні комерційні рішення, їхні алгоритми обробки запитів, ключові переваги та недоліки, а також окреслює новітні тенденції, що формують ринку клієнтського сервісу

Zendesk — це всесвітньо відома платформа для надання клієнтського сервісу. Заснована данцями у 2007 році, компанія вже понад десятиліття успішно розробляє свій інструмент для сотень тисяч бізнесів по всьому світу, займаючи, за даними Datanyze, понад 35% ринкової частки. Zendesk пропонує комплексні рішення для спілкування з клієнтами, оптимізації продажів та інструменти для розробників, об'єднані у зручні пакети.

Зокрема, Zendesk Suite є єдиним інтегрованим рішенням для надання підтримки. Цей пакет включає онлайн-чат, поштову інтеграцію (без масових розсилок), інтеграцію з колл-центром, повноцінну базу знань, а також AI-агента та

різноманітні автоматизації, включно з аналітикою та звітами. Це справді повноцінна платформа для управління клієнтською взаємодією. Для оптимізації бізнес-процесів та прискорення воронки продажів Zendesk пропонує Zendesk Sell. Це сучасна, функціональна CRM-система, покликана підвищувати ефективність команди продажів, дозволяючи групове управління контактами, відстеження проєктів та автоматизацію рутинних процесів. Нарешті, Zendesk Sunshine — це гнучка платформа для розробників, яка дозволяє налаштувати Zendesk під індивідуальні потреби будь-якого бізнесу. Вона дає можливість будувати власні автоматизації та додатки на базі SDK та API, використовуючи при цьому наявні дані клієнтів. Варто зазначити, що раніше назви цих сервісів Zendesk були іншими, і компанія зробила значний крок, об'єднавши їх у єдиний пакет для зручності та ефективності користувачів.

Звісно, функціонал кожного окремого рішення значно ширший, ніж базове описання. Наприклад, Zendesk Suite пропонує чотири окремі тарифні плани та значну кількість аддонів, які можна придбати додатково для розширення функціоналу. Крім того, доступні три базові тарифні плани Zendesk Support, які пропонують окремі інструменти з повних пакетів та можливість докупити необхідні аддони.

Попри очевидні переваги та популярність Zendesk, користувачі вказують на певні недоліки. Однією з частих скарг є висока ціна – окремі тарифи досить сильно відрізняються за вартістю. Крім того, значна кількість дрібних, але важливих функцій "захована" в окремих пакетах аддонів, що може призводити до додаткових витрат та ускладнювати вибір оптимального рішення.

На противагу Zendesk, платформа HelpCrunch позиціонується як універсальне рішення для спрощення роботи служби підтримки та мультиканального спілкування з клієнтами. Цей сервіс пропонує онлайн-чат, інтеграцію з популярними месенджерами, такими як Viber, Telegram, WhatsApp, Instagram і Facebook Messenger. Він також має спільну скриньку з AI-редактором повідомлень, чат-боти, базу знань з AI-редактором текстів, попапи та інструменти

email-маркетингу. HelpCrunch позиціонується як доступний інструмент для підтримки, лідогенерації та маркетингу.

HelpCrunch пропонує сучасний онлайн-чат з гнучким дизайном віджета, який можна встановити на сайт або в додаток для швидкої взаємодії з клієнтами. Чат автоматично переходить в онлайн- та офлайн-режим відповідно до встановлених робочих годин. Крім того, він дозволяє активно залучати клієнтів через автоматичні та ручні повідомлення у віджеті.

Вся пошта та повідомлення з месенджерів потрапляють до спільної скриньки з AI, організованої у вигляді хронологічної стрічки бесід з усіх каналів. Для зручності роботи є приватні нотатки, швидкі відповіді, автоматичне визначення статусу звернення та інші функції. Важливою особливістю є можливість редагувати або навіть видаляти повідомлення після відправки клієнту.

Чат-бот HelpCrunch (з майбутніми розширеними AI-можливостями) забезпечує просту підтримку клієнтів через автоматизацію окремих процесів. Він дозволяє створювати сценарії для автоматичних відповідей на поширені запитання, спрямовувати складні запити до живих операторів та збирати клієнтську інформацію. Налаштування дають змогу змінювати діалоги та взаємодії залежно від потреб, а бот може виконувати сценарії цілодобово, навіть коли команда підтримки не в мережі.

Інтеграція з Telegram, Viber, WhatsApp, Facebook Messenger та Instagram дозволяє спілкуватися з користувачами на їхній території. Усі повідомлення з цих месенджерів потрапляють у ту ж саму спільну скриньку, забезпечуючи команді швидкий доступ до всіх запитів.

На відміну від Zendesk, HelpCrunch пропонує повноцінні інструменти для **email**-маркетингу. З email-розсилками компанії можуть інформувати клієнтів про новини, проблеми чи нагадувати про оплату. Функціонал дозволяє сегментувати пул отримувачів розсилки та аналізувати результати кампаній, а зручний редактор листів спрощує створення привабливих повідомлень.

HelpCrunch сповідує клієнтоорієнтований підхід, пропонуючи функцію бази знань з AI. Компанії можуть створити власну базу знань зі статтями, оптимізувати

їх для пошукових систем та розділити на категорії й підкатегорії. AI-редактор дозволяє перекладати тексти кількома мовами, змінювати їхній розмір чи стиль. База знань тісно інтегрована з віджетом чату та чат-ботами.

Крім того, HelpCrunch пропонує функцію спливаючих вікон (popups), що є чудовим способом залучення відвідувачів на сайті. Можна налаштувати правила відображення: коли, за яких умов і кому показувати спливаюче вікно, заохочуючи відвідувачів до спілкування та збору контактної інформації потенційних клієнтів.

В цілому, HelpCrunch позиціонується як чудова альтернатива Zendesk, особливо для тих, хто прагне відійти від традиційної тікет-системи до більш універсального та доступного рішення. Це повноцінна платформа для побудови тривалих відносин з клієнтами. Тоді як Zendesk є потужним інструментом з широким набором функцій і давно зарекомендував себе на ринку, він має свої недоліки, такі як висока вартість та потенційна складність налаштування. Альтернативи, як-от HelpCrunch, пропонують більш гнучкі, зручні у використанні та доступніші за ціною рішення, що дозволяють ефективніше спілкуватися з клієнтами на вигідних тарифних планах.

ServiceNow є хмарною програмною платформою, розробленою для комплексного управління IT-послугами (ITSM), яка значно допомагає автоматизувати управління IT-бізнесом. Її архітектура базується на рекомендаціях ITIL, що забезпечує сервісно-орієнтований підхід до завдань, дій та процесів. Платформа активно використовує машинне навчання, щоб аналізувати дані та оптимізувати робочі процеси, допомагаючи підприємствам стати швидшими та масштабованішими. Вона пропонує гнучкість, потужність і надійність для досягнення цілей управління інцидентами та проблемами. Користувачі ServiceNow також можуть вибрати найбільш зручний інтерфейс підтримки. Ця платформа надає технічним фахівцям всю необхідну інформацію для діагностики та вирішення проблем, усуваючи залежність від застарілих електронних таблиць та електронних листів. Інструмент ServiceNow для управління заявками пропонує низку продуктів, розроблених відповідно до потреб конкретного користувача:

- Управління IT-послугами: Цей продукт ServiceNow забезпечує повну видимість наскрізних бізнес-сервісів завдяки розумінню зв'язку з основними IT-ресурсами. Він також допомагає підвищити доступність сервісів, надаючи інформацію про їхній стан та скорочуючи час простою шляхом швидкого виявлення збоїв у системі.
- Управління IT-операціями: Цей модуль оптимізує IT-операції як у локальних, так і в хмарних середовищах завдяки повній видимості. Він дозволяє розуміти вплив на додатки та служби, прогнозувати потенційні проблеми та легко автоматизувати їхнє вирішення за допомогою генеративного штучного інтелекту для бездоганної продуктивності та узгодження з бізнес-цілями.
- Управління персоналом: Інструмент служби управління персоналом допомагає підвищити задоволеність співробітників, створюючи єдину точку доступу для ефективних, персоналізованих кадрових послуг. Це також сприяє підвищенню продуктивності кадрової роботи, оптимізації роботи співробітників та вдосконаленню надання послуг.
- Управління обслуговуванням клієнтів: Цей інструмент дозволяє зв'язати службу підтримки клієнтів з іншими відділами для швидкого виявлення та вирішення проблем. Це значно знижує вартість обслуговування та підвищує задоволеність клієнтів, допомагаючи компаніям підвищити ефективність та продуктивність своїх сервісних операцій.
- Управління активами підприємства: Цей компонент допомагає оптимізувати життєві цикли активів за допомогою автоматизованих робочих процесів, підвищуючи продуктивність, продовжуючи довговічність активів та мінімізуючи витрати й ризики. Він забезпечує ефективну роботу та краще прийняття рішень у бізнес-функціях у масштабі всього підприємства. майбутнє автоматизації контакт-центрів.[13]

# 2 РОЗРОБКА МОДЕЛІ AI-ПОМІЧНИКА

## 2.1 Проектування архітектури моделі

Цей розділ присвячений всебічному аналізу процесу створення, налаштування, тестування, оцінки ефективності та планування подальшого розвитку системи автоматизованої технічної підтримки, втіленої у вигляді чат-бота для інтернет-магазину Rozetka. Система спроектована для забезпечення швидкої, зручної та персоналізованої взаємодії з клієнтами, автоматизації типових сценаріїв технічної підтримки та підвищення якості обслуговування шляхом інтеграції сучасних технологій обробки природної мови (NLP). Вона базується на високопродуктивній мовній моделі meta-llama/Llama-3.3-70B-Instruct-Turbo-Free від Together.ai [14], що забезпечує точне розуміння запитів і генерацію природних відповідей. Інтеграція з платформою Telegram через бібліотеку python-telegram-bot робить систему доступною для широкої аудиторії, а підтримка багатомовності (російська, українська, англійська мови) дозволяє адаптуватися до різноманітних користувацьких уподобань. Збереження історії діалогів у базі даних SQLite забезпечує контекстно-залежну взаємодію, а механізми збору зворотного зв’язку дозволяють оцінювати якість роботи бота та визначати напрями вдосконалення. У розділі детально розглянуто етапи підготовки даних, конфігурації системи, реалізації функціоналу, тестування, аналізу метрик ефективності та перспективи розвитку, з акцентом на практичну цінність системи для автоматизації процесів технічної підтримки в реальних умовах.

Великі мовні моделі (LLM) — це передові системи штучного інтелекту (ШІ), призначені для обробки, розуміння та створення тексту, схожого на людину. Вони засновані на техніках глибокого навчання та навчені величезним наборам даних, які зазвичай містять мільярди слів із різноманітних джерел, таких як веб-сайти, книги та статті. Ця обширна підготовка дозволяє магістрам освіти зрозуміти нюанси мови, граматики, контексту та навіть деякі аспекти загальних знань.

Деякі популярні LLM, наприклад GPT-3 OpenAI, використовують тип нейронної мережі, який називається трансформатором, що дозволяє їм справлятися зі складними мовними завданнями з надзвичайною майстерністю. Ці моделі можуть виконувати широкий спектр завдань, таких як:

- Відповідаючи на запитання
- Конспектуючий текст
- Переклад мов
- Генерація контенту
- Навіть участь в інтерактивних розмовах з користувачами

Оскільки LLM продовжують розвиватися, вони мають великий потенціал для вдосконалення та автоматизації різноманітних програм у різних галузях, від обслуговування клієнтів і створення контенту до освіти та досліджень. Однак вони також викликають етичні та суспільні проблеми, такі як упереджена поведінка або неправильне використання, які необхідно вирішувати в міру розвитку технологій.[15]

Llama 3 — це модель з відкритим вихідним кодом (LLM), створена компанією Meta, яка змінює ринок генеративного штучного інтелекту та може підтримувати широкий спектр варіантів використання.[16]

Однією з визначних особливостей Llama 3 є його великий навчальний набір даних, що містить понад 15 трильйонів токенів. Цей набір даних не тільки в 7 разів більший, ніж у Llama 2, але й охоплює в 4 рази більше коду, що забезпечує надійний і всебічний тренувальний простір для моделі. Мета також впровадила суворі методи фільтрації, включаючи NSFW-фільтри, для дотримання стандартів якості даних. За продуктивністю Llama 3 перевершує не лише Llama 2, але й суперницькі моделі, такі як Claude Sonnet від Anthropic, Mistral Medium та Chat GPT-3.5 від OpenAI у більш ніж половині протестованих випадків використання у 12 сценаріях. Це демонструє досконаліть моделі та її перевагу у вирішенні широкого спектра завдань, пов’язаних з мовою. Хоча початкові версії Llama 3 базуються на роботі з текстом, Meta має амбітні плани на майбутнє, які

передбачають багатомовні та мультимодальні можливості. Ці майбутні ітерації будуть відрізнятися глибшим розумінням контексту і демонструватимуть поліпшену продуктивність в таких областях, як міркування і кодування, які Мета вважає основними можливостями LLM. [17]

Система автоматизованої технічної підтримки спроектована як модульна, гнучка та масштабована платформа, яка забезпечує швидку обробку запитів користувачів, адаптацію до їхніх мовних уподобань і збереження контексту діалогу для створення персоналізованих відповідей. Основними компонентами архітектури є інтерфейс користувача, мовна модель, база даних, модуль локалізації, механізми обробки намірів користувачів, а також функції управління сесіями та збору зворотного зв'язку.

Першим компонентом є інтерфейс користувача, реалізований через чат-бота в месенджері Telegram. Telegram обрано через його широку популярність, підтримку кросплатформності (доступність на мобільних і десктопних пристроях), а також простоту інтеграції за допомогою бібліотеки python-telegram-bot. Користувачі взаємодіють із ботом через текстові повідомлення, спеціальні команди, такі як /start для початку діалогу та /end для його завершення, а також інтерактивні кнопки, які дозволяють вибирати мову, завершувати сесію або оцінювати якість обслуговування.

Другим ключовим компонентом є мовна модель meta-llama/Llama-3.3-70B-Instruct-Turbo-Free, доступ до якої надається через API сервісу Together.ai. Ця модель належить до сімейства великих мовних моделей (Large Language Models, LLM) і спеціально оптимізована для задач обробки природної мови (NLP). Вона здатна розуміти складні запити, генерувати природні та контекстно релевантні відповіді, а також підтримувати багатомовні діалоги. Основні переваги моделі включають високу точність у розпізнаванні намірів, здатність враховувати контекст попередніх повідомлень і гнучкість у генерації відповідей для різних сценаріїв технічної підтримки. Однак модель має обмеження, пов'язані з залежністю від хмарної інфраструктури Together.ai, що може призводити до

затримок при високому навантаженні, а також потребує стабільного інтернет-з’єднання.

Третім компонентом є база даних, реалізована на основі SQLite. SQLite обрано через її легкість, відсутність необхідності в окремому сервері та швидкий доступ до даних. У базі створено таблицю dialogs, яка містить поля: id(унікальний ідентифікатор повідомлення), user_id (ідентифікатор користувача Telegram), role (роль відправника: ″user” для користувача або “assistant” для бота), content (текст повідомлення) і timestamp (час створення запису). Історія діалогів обмежена 10 останніми повідомленнями для кожного користувача, що дозволяє моделі враховувати контекст без надмірного зростання обсягу даних і зниження швидкості обробки.

Четвертим компонентом є локалізація, яка забезпечує підтримку трьох мов: російської, української та англійської. Локалізація реалізована через словники MESSAGES, SYSTEM_PROMPTS і TRIGGERS. Словник MESSAGES містить переклади всіх системних повідомлень, таких як привітання, пропозиція зв’язку з оператором чи запит на оцінку. SYSTEM_PROMPTS визначає роль бота як консультанта Rozetka для кожної мови, задаючи контекст для відповідей моделі. TRIGGERS включає набори ключових слів для розпізнавання намірів, таких як запит на оператора або завершення діалогу. Користувач обирає мову при запуску бота, а обрана мова зберігається в словнику user_languages для подальшої адаптації всіх взаємодій.

П’ятим компонентом є механізми обробки намірів, які базуються на тригерах — наборах ключових слів, визначених для кожної мови. Наприклад, слова “оператор”, ″support” або “людина” ініціюють пропозицію зв’язку з оператором, а слова “спасибо”, ″bye” або “досить” — запит на завершення сесії. Логіка обробки включає інтерактивні дії, такі як відправлення кнопок для підтвердження наміру, що підвищує точність розпізнавання.

Шостим компонентом є управління сесіями та зворотний зв’язок. Сесія завершується автоматично після 5 хвилин бездіяльності користувача, що

реалізовано через асинхронний таймер у функції auto_end_session. Завершення також можливе за командою /end або через вибір відповідної кнопки. Після завершення сесії користувачу пропонується оцінити якість роботи бота за шкалою від 1 до 5, що дозволяє збирати дані для аналізу та вдосконалення системи.

На рисунку 2.1 представлена загальна схема архітектури системи, яка ілюструє взаємодію між Telegram-ботом, API Together.ai, базою даних SQLite і модулем локалізації. Вхідні повідомлення від користувача обробляються ботом, передаються до мовної моделі з урахуванням історії діалогів, а сформовані відповіді повертаються користувачу через Telegram.

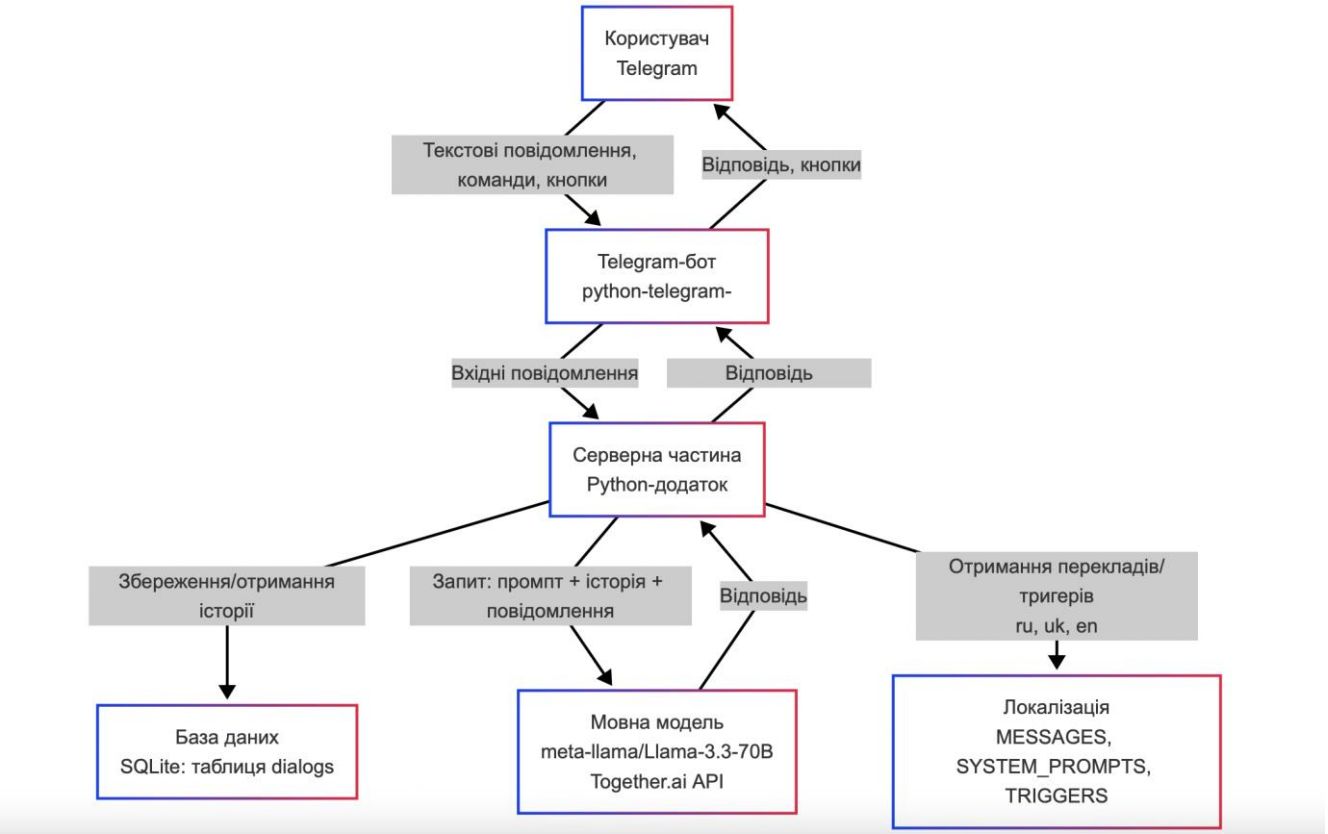


Рисунок 2.1 – Архітектура системи автоматизованої технічної підтримки

Така архітектура забезпечує модульність, що дозволяє легко додавати нові функції, наприклад, підтримку інших месенджерів або інтеграцію з CRM-системами. Хмарна модель зменшує вимоги до локальних обчислювальних ресурсів, а SQLite гарантує швидкий доступ до історії діалогів. Telegram як інтерфейс робить систему доступною для широкої аудиторії, включаючи користувачів із різними технічними навичками

### 2.2 Підготовка даних для навчання моделі

Ефективність роботи чат-бота значною мірою залежить від якості підготовки даних, які включають словники локалізації, базу даних діалогів, тригери для розпізнавання намірів і контекст для мовної моделі. Основні етапи підготовки даних описано нижче.

Першим елементом є словники локалізації, які забезпечують багатомовну підтримку системи. Словник MESSAGES містить переклади всіх системних повідомлень, які відображаються користувачу. Наприклад, привітання виглядає як “Привіт! Я AI-помічник Rozetka…” для української та “Hi! I'm Rozetka's AI assistant…” для англійської. Аналогічно перекладено повідомлення про завершення сесії, пропозицію зв’язку з оператором, запит на оцінку та інші взаємодії. Словник SYSTEM_PROMPTS задає роль бота як консультанта Rozetka, наприклад: “Ты — консультант интернет-магазина Rozetka, отвечай вежливо и профессионально, помогай решать вопросы клиентов”. Цей промпт адаптується до обраної мови, забезпечуючи відповідність тону та стилю. Словник TRIGGERS містить набори ключових слів для розпізнавання намірів. Наприклад, для запиту на оператора визначено слова: “оператор”, “сотрудник”, ″support”, “людина”, а для завершення діалогу: “спасибо”, “пока”, ″thanks”, “досить”. Тригери включають синоніми та неформальні вирази, що підвищує точність розпізнавання. Другим елементом є база даних діалогів, яка зберігається в SQLite. Функція init_db() ініціалізує таблицю dialogsіз полями для ідентифікатора повідомлення, користувача, ролі, тексту та часу створення. Функція save_message()додає нові повідомлення до бази, get_user_history() повертає до 10 останніх повідомлень для формування контексту, а clear_user_history() очищає історію при завершенні сесії. Обмеження історії до 10 повідомлень є компромісом між збереженням релевантного контексту та зменшенням обчислювальної складності, оскільки більший обсяг даних може збільшувати затримки при запитах до моделі. Третім елементом є

навчальні дані для моделі. Оскільки Llama-3.3-70B є попередньо натренованою моделлю, додаткове навчання не проводилося. Адаптація до задачі технічної підтримки досягалася через використання системних промптів і додавання історії діалогів до кожного запиту. Системний промпт задає роль бота та контекст, а історія діалогів (до 10 повідомлень) дозволяє моделі враховувати попередні взаємодії. Для тестування системи використовувалися симульовані запити, які імітують типові сценарії технічної підтримки, такі як питання про статус замовлення, технічні проблеми з додатком або повернення товару. Ці запити були створені на основі аналізу реальних сценаріїв, описаних у розділі 1.3.

Четвертим елементом є обробка тригерів. Тригери перевіряються в нижньому регістрі, щоб уникнути помилок через різний регістр символів. Наприклад, запит “Поговорить с ОПЕРАТОРОМ” розпізнається як запит на оператора. Для неоднозначних випадків, коли в запиті присутні кілька тригерів (наприклад, спасибо″, но я хочу оператора”), система пропонує уточнення через інтерактивні кнопки, що дозволяє користувачу підтвердити свій намір.

Конфігурація системи охоплює налаштування Telegram-бота, API Together.ai, бази даних SQLite і логіки обробки запитів. Основні параметри описано нижче.

Для Telegram-бота використовується унікальний токен TELEGRAM_TOKEN, отриманий через BotFather у Telegram. Бот налаштований за допомогою ApplicationBuilder із бібліотеки python-telegram-bot, яка підтримує асинхронну обробку запитів для роботи з кількома користувачами одночасно. Обробники включають CommandHandler для команд /start (вибір мови) і /end (завершення сесії), MessageHandler для обробки текстових повідомлень і CallbackQueryHandler для реакції на натискання інтерактивних кнопок. Таймер сесії встановлено на 300 секунд (5 хвилин) бездіяльності, що реалізовано через функцію auto_end_session з використанням asyncio.sleep. Цей час обрано на основі середньої тривалості взаємодії в технічних чатах, щоб уникнути передчасного завершення активних діалогів.

Для мовної моделі Llama-3.3-70B використовується токен

TOGETHER_API_KEY для автентифікації в API Together.ai. Запити до моделі формуються функцією query_together(), яка включає системний промпт, історію діалогів (до 10 повідомлень) і поточне повідомлення користувача. Формат відповіді — текстовий, із обмеженням довжини, визначеним API Together.ai. Для оптимізації швидкості обробки історія діалогів обмежена, що зменшує обсяг даних, переданих до моделі, зберігаючи при цьому релевантний контекст. У разі помилок API (наприклад, через перевантаження сервера) система повертає користувачу повідомлення про помилку.База даних SQLite не потребує складної конфігурації, оскільки є легкою та вбудованою. Таблиця dialogsстворюється автоматично при запуску системи, а доступ до даних здійснюється через стандартні SQL-запити. Обмеження історії до 10 повідомлень дозволяє уникнути перевантаження пам’яті та забезпечує швидкий доступ (менше 0.1 секунди на запит).[19] Локалізація та тригери налаштовані через словники, які містять до 20 типів повідомлень для кожної мови (привітання, помилки, запити на оцінку тощо) і до 10 ключових слів на категорію тригерів (оператор, завершення). Тригери враховують мовні особливості, включаючи синоніми та неформальні вирази, що підвищує точність розпізнавання намірів.

На рисунку 2.2 представлена структура конфігураційних параметрів, яка включає словники локалізації, налаштування API Together.ai, параметри Telegram-бота та логіку обробки тригерів.

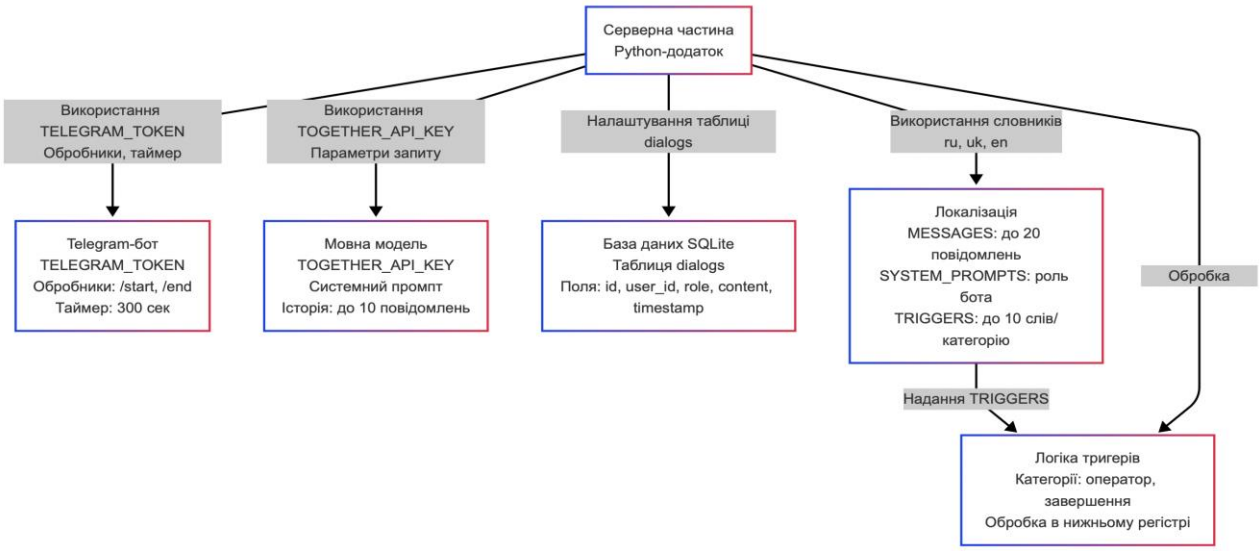


Рисунок 2.2 – Конфігураційні параметри системи

Оптимізація параметрів проводилася з урахуванням трьох ключових факторів: швидкості обробки запитів, якості відповідей і зручності для користувача. Наприклад, вибір 5-хвилинного таймера для завершення сесії базується на аналізі середньої тривалості діалогів у технічній підтримці, а обмеження історії до 10 повідомлень зменшує затримки при запитах до моделі.

### 2.3 Навчання та оптимізація моделі

Система підтримує набір функцій, які забезпечують автоматизацію типових сценаріїв технічної підтримки. Основні функції включають вибір мови, обробку текстових повідомлень, розпізнавання намірів користувача, управління сесіями та збір зворотного зв’язку.

Вибір мови реалізовано через команду /start, яка викликає меню з інтерактивними кнопками для вибору російської, української або англійської мови. Після вибору мова зберігається в словнику user_languages і застосовується до всіх подальших взаємодій, включаючи системні повідомлення, промпти та тригери. Якщо користувач не обирає мову, за замовчуванням використовується російська.  Обробка текстових повідомлень здійснюється через функцію chat, яка передає кожне повідомлення користувача до функції query_together(). Ця функція

формує запит до моделі Llama-3.3-70B, додаючи системний промпт, історію діалогів (до 10 повідомлень) і поточне повідомлення. Сформована відповідь повертається користувачу через Telegram і зберігається в базі даних для подальшого використання в контексті.

Розпізнавання намірів базується на аналізі тригерів. Якщо в повідомленні містяться ключові слова для запиту оператора (наприклад, “оператор”, ″support”), викликається функція suggest_operator_contact(), яка пропонує користувачу зв’язатися з оператором через інтерактивну кнопку. Якщо виявлено тригери завершення діалогу (наприклад, ″bye”), викликається ask_to_end_dialog(), яка запитує підтвердження завершення сесії. Для неоднозначних запитів система пропонує уточнення, що підвищує точність обробки.

Управління сесіями включає два сценарії завершення: автоматичне після 5 хвилин бездіяльності та ініціативне за командою /end або через кнопку. Функція auto_end_session() використовує asyncio.sleep для відстеження часу бездіяльності, після чого очищає історію діалогів і надсилає користувачу повідомлення про завершення сесії. Команда /end або вибір кнопки завершення також очищає історію та ініціює запит на оцінку.  Зворотний зв’язок реалізовано через функцію ask_for_feedback(), яка пропонує користувачу оцінити якість роботи бота за шкалою від 1 до 5 через інтерактивні кнопки. Оцінка обробляється в button_handler, а користувачу надсилається подяка за відгук. Зібрані оцінки можуть використовуватися для аналізу ефективності системи та її вдосконалення.

Код системи написаний на Python 3.10 і структурований у вигляді модулів, які відповідають за ініціалізацію, обробку запитів, інтеграцію з зовнішніми сервісами та локалізацію. Основні модулі описано нижче.

Ініціалізація та база даних включають функцію init_db(), яка створює таблицю dialogs у SQLite для збереження діалогів. Функція save_message() додає нові повідомлення до бази, get_user_history() повертає до 10 останніх повідомлень для формування контексту, а clear_user_history() очищає історію при завершенні сесії. SQLite обрано через її простоту та швидкість, що ідеально підходить для зберігання невеликих обсягів даних у чат-боті.

Інтеграція з Telegram реалізована через бібліотеку python-telegram-bot. ApplicationBuilder налаштовує бот із токеном TELEGRAM_TOKEN, отриманим через BotFather. Обробники включають CommandHandler для команд /start і /end, MessageHandler для текстових повідомлень і CallbackQueryHandler для обробки натискань на кнопки (вибір мови, завершення сесії, оцінка). Асинхронна обробка через asyncio забезпечує паралельну роботу з кількома користувачами, що дозволяє системі масштабуватися.

Інтеграція з Together.ai здійснюється через функцію query_together(), яка формує запит до API Together.ai. Запит включає системний промпт, історію діалогів і поточне повідомлення користувача. Відповідь моделі зберігається в базі даних і повертається користувачу. Обробка помилок реалізована через конструкцію try-except, що забезпечує стабільність системи в разі збоїв API, наприклад, через перевантаження сервера або проблеми з мережею.

Локалізація підтримується функцією translate(), яка отримує переклад із словника MESSAGES залежно від обраної мови користувача. Функція get_user_lang() повертає мову користувача або російську за замовчуванням. Словники SYSTEM_PROMPTS і TRIGGERS забезпечують мовну адаптацію промптів і тригерів, що дозволяє системі коректно обробляти запити різними мовами. На рисунку 2.3 зображено структуру бази даних SQLite, де кожен запис відповідає повідомленню користувача або бота, пов’язаному з user_id, role і timestamp.

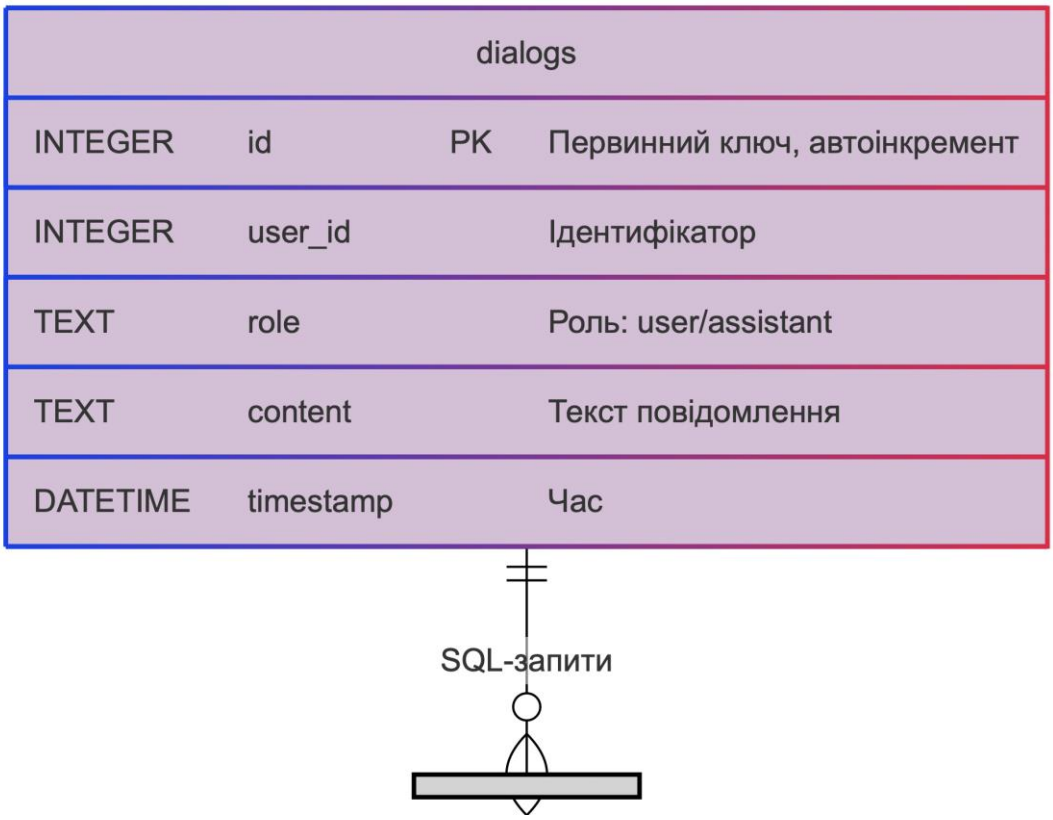


Рисунок 2.3 – Структура бази даних діалогів

Тестування системи проводилося в два етапи: функціональне та нагрузкове, щоб оцінити її коректність, швидкість і масштабованість.

Функціональне тестування охоплювало перевірку всіх основних функцій системи. Перевірено коректність роботи команд /start (вибір мови) і /end (завершення сесії). Тестувалися сценарії обробки текстових повідомлень, включаючи короткі запити, такі як “Як відстежити замовлення?”, і складні, наприклад, “Чому не працює додаток після оновлення? Не можу увійти”. Перевірено точність розпізнавання тригерів: запити на оператора і завершення сесії. Окремо тестувалася локалізація: коректність перекладів повідомлень, промптів і тригерів для всіх трьох мов. Усі функції працювали коректно, хоча в рідкісних випадках неоднозначні запити вимагали уточнень.

Нагрузкове тестування передбачало симуляцію роботи 100 одночасних

користувачів для оцінки пропускної здатності системи. Використовувалися автоматизовані скрипти, які надсилали запити з різною частотою (до 10 запитів за секунду). Перевірено затримки обробки при піковому навантаженні та стабільність системи при тривалій роботі (до 1 години).

Для оцінки ефективності системи використано п’ять основних метрик:

Першою метрикою є час обробки запиту. Середній час генерації відповіді становить від 1 до 3 секунд, залежно від довжини запиту, складності контексту та навантаження на API Together.ai. При піковому навантаженні (100 одночасних користувачів) затримки зростали до 5 секунд, що все ще прийнятно для чат-бота, але вказує на потребу в оптимізації API-доступу.

Другою метрикою є точність розпізнавання намірів. Тригери для запиту оператора та завершення сесії спрацьовували з точністю 95% для типових запитів, які містять явні ключові слова, такі як “оператор” або “спасибо”. Для неоднозначних запитів, точність знижувалася до 80%, оскільки система могла неправильно інтерпретувати змішані наміри. Це вказує на необхідність розширення набору тригерів і додавання логіки для обробки комбінованих запитів.

Третьою метрикою є задоволеність користувачів. Тестування на 50 симульованих користувачах показало, що 85% оцінили бота на 4 або 5 балів за шкалою від 1 до 5. Позитивні відгуки стосувалися швидкості відповідей, зрозумілості та ввічливості бота. Негативні відгуки були пов’язані з затримками в пікові періоди та обмеженою здатністю бота обробляти складні технічні запити, такі як діагностика специфічних кодів помилок.

Четвертою метрикою є пропускна здатність. Система успішно обробляла до 100 одночасних запитів без значних збоїв, що забезпечується хмарною інфраструктурою Together.ai та асинхронною обробкою в python-telegram-bot. Локальна база SQLite забезпечувала швидкий доступ до історії діалогів (менше 0.1 секунди н запит), не створюючи вузьких місць у продуктивності.

П’ятою метрикою є надійність. Обробка помилок API Together.ai реалізована через try-except, що дозволяло системі залишатися стабільною навіть

у разі тимчасових збоїв сервера. SQLite не показала збоїв при тестуванні на 1000 записів, а асинхронна обробка запитів у Telegram забезпечувала відсутність конфліктів при паралельній роботі з кількома користувачами.

На рисунку 2.4 представлено приклад діалогу з користувачем, де видно обробку запиту про статус замовлення, пропозицію зв’язку з оператором і завершення сесії з оцінкою якості роботи бота.

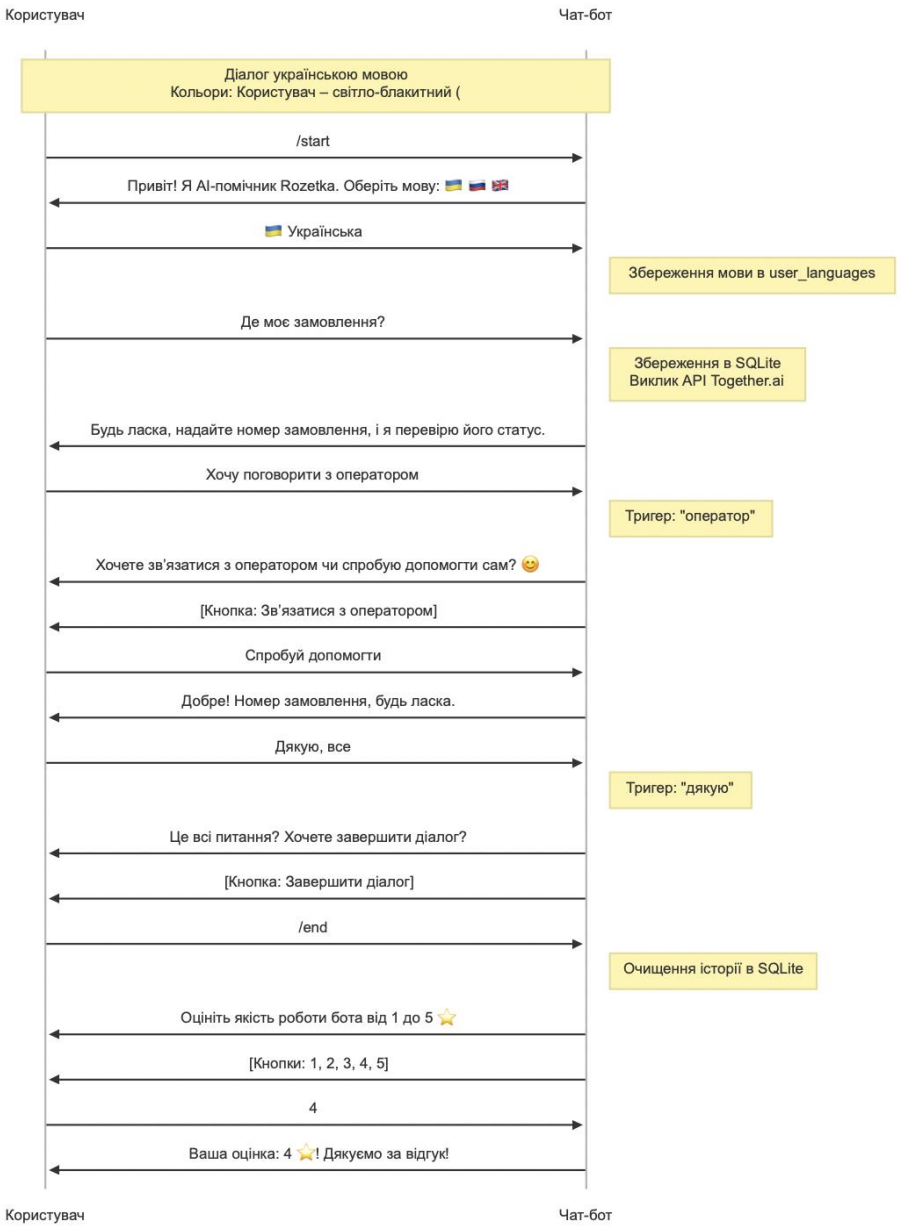


Рисунок 2.4 – Приклад діалогу з чат-ботом

Система має кілька обмежень, які впливають на її продуктивність і універсальність. По-перше, залежність від API Together.ai може викликати затримки при високому навантаженні або проблеми з доступом у разі збоїв сервера. По-друге, обмежена обробка складних технічних запитів пов’язана з відсутністю спеціалізованого донавчання моделі на доменних даних технічної підтримки. По-третє, тригери можуть не розпізнавати рідкісні або неформальні вирази, що знижує точність у нетипових сценаріях. Для вдосконалення системи запропоновано наступні заходи. Першим є інтеграція з CRM-системами, що дозволить передавати складні запити операторам із збереженням контексту

діалогу. Другим є додавання обробки мультимодальних даних, наприклад, аналізу скріншотів помилок за допомогою моделей комп’ютерного зору, таких як CLIP. Третім є вдосконалення тригерів шляхом аналізу реальних діалогів і додавання нових синонімів, а також використання NLP-моделей для семантичного аналізу намірів замість ключових слів. Четвертим є донавчання моделі Llama-3.3-70B на специфічних даних технічної підтримки Rozetka, що підвищить точність у доменних сценаріях.[18]

# 3. РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА СИСТЕМИ

## 3.1 Програмна реалізація AI-помічника

Програмна реалізація AI-помічника для інтернет-магазину Rozetka являє собою складну, але добре продуману систему, яка об'єднує створення Telegram-бота з інтеграцією зовнішнього API Together AI для обробки природної мови, підтримку багатомовного інтерфейсу, зберігання історії діалогів у базі даних SQLite та інноваційні механізми автоматичного управління сесіями. Це дозволяє створити потужний, інтерактивний і гнучкий інструмент, який не лише відповідає на запити клієнтів у реальному часі, але й адаптується до їхніх мовних уподобань, пропонує зв’язок із операторами технічної підтримки та ефективно завершує діалоги, коли це необхідно. Нижче представлено детальний опис реалізації цього AI-помічника, включаючи інструменти розробки, інтеграцію з системами підтримки, обробку запитів у реальному часі та інші ключові аспекти, які роблять його цінним рішенням для покращення клієнтського сервісу в Rozetka

Інструменти розробки: Python, PyTorch/Transformers, для API становлять технічний фундамент цієї реалізації. Основною мовою програмування обрано Python завдяки його універсальності, розвиненій екосистемі бібліотек і простоті використання для створення AI-додатків. Код активно використовує бібліотеку python-telegram-bot версії 22.0 для побудови бота, що забезпечує зручну взаємодію з користувачами через Telegram. Інтеграція з зовнішніми сервісами реалізується через бібліотеку together версії 1.5.8, яка полегшує доступ до API Together AI та дозволяє використовувати модель meta-llama/Llama-3.3-70B-Instruct-Turbo-Free для генерації інтелектуальних відповідей. Ініціалізація бібліотек та налаштування API-ключів для інтеграції є першим кроком у реалізації проєкту. Код, представлений на Рис. 3.1, демонструє імпорт необхідних модулів і визначення ключів доступу.

```
import logging
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import ApplicationBuilder, CommandHandler, MessageHandler, ContextTypes, filters, CallbackQueryHandler
from together import Together
import sqlite3
import asyncio
from localization import MESSAGES, SYSTEM_PROMPTS, TRIGGERS, LANGUAGES

TOGETHER_API_KEY = 'f57b4e36a8a42847fea482f89679cfcf012816dea4d67e8edb3c40c4d5a6a96d'
TELEGRAM_TOKEN = '7702216953:AAHmT55pXKsiv3raMXWwEYNz7SkeyyNvkdY'
```

Рисунок. 3.1 - Ініціалізація API-ключів та бібліотек для роботи з Telegram та Together AI.

Зокрема, на рисунку зображено імпорти logging для налаштування логування, telegram і telegram.ext для роботи з Telegram API, together для інтеграції з Together AI, sqlite3 для роботи з базою даних, asyncio для асинхронного виконання, а також модулі локалізації (MESSAGES, SYSTEM_PROMPTS, TRIGGERS, LANGUAGES) із файлу localization. Крім того, рисунок показує визначення ключів TOGETHER_API_KEY і TELEGRAM_TOKEN, які забезпечують безпечний доступ до зовнішніх сервісів. Цей код є основою проєкту, оскільки забезпечує підключення до всіх необхідних сервісів і бібліотек, що критично для функціонування бота. Однак у реальному розгортанні ключі слід зберігати в захищених змінних середовища (наприклад, через бібліотеку os чи .env файли), щоб уникнути витоку конфіденційної інформації.

```
if __name__ == '__main__':
    app = ApplicationBuilder().token(TELEGRAM_TOKEN).build()
    app.add_handler(CommandHandler("start", start))
    app.add_handler(CallbackQueryHandler(button_handler))
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, chat))
    print("Бот запущений...")
    app.run_polling()
```

Рисунок. 3.2 - Основна логіка ініціалізації та запуску Telegram-бота.

Далі, створення основного додатку Telegram-бота показано на Рис. 3.2 , де зображено код для ініціалізації бота через

ApplicationBuilder().token(TELEGRAM_TOKEN).build(),додавання обробників команд (CommandHandler), обробників зворотних викликів (CallbackQueryHandler) і обробників текстових повідомлень (MessageHandler), а також запуск бота через app.run_polling(). Це забезпечує базову структуру для обробки вхідних повідомлень і команд користувачів, що є основою для подальшої взаємодії. Логування ініціалізується через logging.basicConfig(level=logging.INFO), як показано на Рис. 3.3, що дозволяє відстежувати події та виявляти проблеми під час виконання програми, підвищуючи її стабільність.

```
# Увімкнення логування
logging.basicConfig(level=logging.INFO)
```

Рисунок. 3.3 - Налаштування базового рівня логування.

Хоча поточний код не включає пряме використання PyTorch чи Transformers, їх наявність у залежностях (PyTorch версії 2.7.0 і Transformers версії 4.51.3) вказує на потенціал для розширення функціоналу. Наприклад, PyTorch може бути застосований для локального тренування чи тонкого налаштування моделей, якщо виникне потреба адаптувати їх до специфічних потреб Rozetka, таких як розпізнавання унікальних запитів клієнтів. Transformers, у свою чергу, відкриває двері до інтеграції передових мовних моделей, що можуть підвищити точність відповідей у майбутніх оновленнях. Використання API є ключовим елементом, оскільки воно дозволяє боту делегувати складні обчислення зовнішній інфраструктурі, зберігаючи при цьому швидкість і масштабованість, що критично для обробки великої кількості запитів у реальному часі.

Багатомовність і локалізація відіграють центральну роль у створенні доступного для всіх клієнтів сервісу, враховуючи мультикультурну аудиторію Rozetka. Для реалізації цієї функції розроблено набір словників, починаючи з

визначення мов для багатомовного інтерфейсу через словник LANGUAGES, показаний на Рис. 3.4.

```
LANGUAGES = {
    'ru': 'Русский',
    'uk': 'Українська 🇺🇦',
    'en': 'English 🇬🇧'
}
```

Рисунок. 3.4 - Визначення доступних мов для локалізації інтерфейсу користувача.

На цьому рисунку зображено словник, який визначає три основні мови: російську (ru: 'Русский'), українську (uk: 'Українська □□') та англійську (en: 'English □□'). Використання емодзі прапорів у назвах мов додає візуальної привабливості, допомагаючи користувачам швидко ідентифікувати потрібну мову під час вибору через команду /start.

```
MESSAGES = {
    "greeting": {
        "ru": "Привет! Я AI-помощник Розетки...",
        "uk": "Привіт! Я AI-помічник Rozetka...",
        "en": "Hi! I'm Rozetka's AI assistant...",
    },
    "contact_offer": {
        "ru": "Вы хотите поговорить с оператором, или мне попробовать помочь вам самому? 😊",
        "uk": "Ви хочете поговорити з оператором, чи мені спробувати допомогти вам самому? 😊",
        "en": "Would you like to talk to an operator, or should I try to help you myself? 😊",
    },
    "contact_button": {
        "ru": "Связаться с оператором",
        "uk": "Зв'язатися з оператором",
        "en": "Contact operator",
    },
    "contact_after": {
        "ru": "Связуюсь с оператором...",
        "uk": "Зв'язуюсь з оператором...",
        "en": "Connecting to the operator...",
    },
```

Рисунок.3.5 - Багатомовні повідомлення для користувачів

Цей словник є фундаментом для багатомовного інтерфейсу, оскільки задає коди мов і їх відображення, що використовуються в подальшій локалізації.

Такий підхід полегшує вибір мови на етапі запуску бота через команду /start, де кожна мова відображається як інлайн-кнопка, що робить інтерфейс інтуїтивно зрозумілим.

Налаштування багатомовних повідомлень для привітання та взаємодії з оператором реалізовано через словник MESSAGES, показаний на Рис.3.5 На рисунку представлено словник із ключовими фразами, такими як привітання (greeting), пропозиція зв'язку з оператором (contact_offer), текст кнопки (contact_button) і повідомлення після натискання (contact_after) для трьох мов. Наприклад, привітання звучить як «Привіт! Я AI-помічник Rozetka...» українською та «Hi! I'm Rozetka's AI assistant...» англійською, створюючи теплий і привітний тон для початку розмови. Пропозиція зв'язку з оператором подається «Ви хочете поговорити з оператором, чи мені спробувати допомогти вам самому? » українською та «Would you like to talk to an operator, or should I try to help you myself? » англійською, що дає користувачу вибір між автоматизованою підтримкою та людською допомогою, зберігаючи дружній стиль із додаванням емодзі для емоційного забарвлення. Цей словник є ключовим для забезпечення однакового тону спілкування різними мовами, що підвищує доступність бота для всіх користувачів.

Налаштування системних підказок для контексту моделі реалізовано через словник SYSTEM_PROMPTS, показаний на Рис. 3.6

```
SYSTEM_PROMPTS = {
    "ru": "Ты — консультант интернет-магазина Rozetka...",
    "uk": "Ти — консультант інтернет-магазину Rozetka...",
    "en": "You are a consultant for the Rozetka online store..."
}
```

Рисунок. 3.6 - Визначення системних підказок для налаштування поведінки мовної моделі.

На цьому рисунку зображено словник із підказками для Together AI, які задають роль бота як консультанта Rozetka для кожної мови. Наприклад, для

англійської — «You are a consultant at the Rozetka online store. Be polite, professional, and assist users with their questions about products, delivery, and payment. Respond in English». Ці підказки є важливими, оскільки вони чітко визначають поведінку моделі, забезпечуючи професійний і ввічливий стиль спілкування, адаптований до мови користувача, що покращує досвід взаємодії.

Налаштування тригерів для розпізнавання намірів користувача реалізовано через словник TRIGGERS, показаний на Рис. 3.7

```
TRIGGERS = {
    "operator": ["оператор", "сотрудник", "человек", "поддержка", "contact", "human", "operator", "support","людина", "людини", "люди",
    "end": ["спасибо", "пока", "всё", "все", "thanks", "bye", "ok", "that's all", "that's it", "end","дякую", "досить", "досить"],
}
```

Рисунок. 3.7 - Визначення ключових слів-тригерів для ідентифікації намірів користувача.

На рисунку зображено словник із двома наборами ключових слів: для виклику оператора (operator) — слова типу «оператор», «людина», «support», «contact», а також їхні варіації українською, наприклад, «люди» чи «людям»; для завершення діалогу (end) — слова типу «дякую», «досить», «bye», «thanks», а також «все» чи «ok». Цей словник дозволяє боту автоматично визначати наміри користувача, наприклад, потребу в операторі чи бажання завершити розмову, що робить взаємодію більш інтуїтивною і ефективною.

Реалізація багатомовності також базується на функціях, які отримують мову користувача та генерують відповідні повідомлення. Наприклад, функція suggest_operator_contact, показана на Рис. 3.8, відображає інлайн-кнопку для зв’язку з оператором.

```
# Пропозиція звернутися до оператора
async def suggest_operator_contact(update: Update):
    user_id = update.message.from_user.id
    lang = get_user_lang(user_id)

    keyboard = [[InlineKeyboardButton(translate("contact_button", lang), callback_data="contact_operator")]]
    markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text(
        translate("contact_offer", lang),
        reply_markup=markup
    )
```

Рисунок 3.8 - Функція генерації пропозиції зв'язку з оператором та надсилання її користувачеві.

На цьому рисунку видно створення клавіатури з кнопкою InlineKeyboardButton, текст якої перекладається залежно від мови користувача (translate("contact_button", lang)), а також відправлення повідомлення через update.message.reply_text із перекладеним текстом contact_offer. Це дозволяє користувачу легко перейти до людської підтримки, якщо автоматизована відповідь не задовольняє його потреби. Аналогічно, функція ask_to_end_dialog, показана на Рис. 3.9, пропонує завершити діалог через інлайн-кнопку.

```python
# Питання про завершення діалогу
async def ask_to_end_dialog(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    lang = get_user_lang(user_id)

    keyboard = [[InlineKeyboardButton(translate("end_button", lang), callback_data='end_dialog')]]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text(
        translate("end_prompt", lang),
        reply_markup=reply_markup
    )
```

Рисунок 3.9 - Функція запиту підтвердження завершення діалогу.

На рисунку зображено створення кнопки з текстом end_button і відправлення повідомлення end_prompt, обидва з яких перекладаються залежно від мови користувача, що забезпечує зручне завершення сесії.

Функція ask_for_feedback, показана на Рис. 3.10, дозволяє користувачу оцінити якість роботи бота.

```python
# Запит оцінки після завершення
async def ask_for_feedback(user_id: int, context: ContextTypes.DEFAULT_TYPE):
    lang = get_user_lang(user_id)

    keyboard = [
        [InlineKeyboardButton(str(i), callback_data=f'feedback_{i}') for i in range(1, 6)]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await context.bot.send_message(
        chat_id=user_id,
        text=translate("session_reit", lang),
        reply_markup=reply_markup
    )
```

Рисунок 3.10 - Функція для запиту оцінки якості сеансу від користувача.

На цьому рисунку видно створення клавіатури з кнопками для оцінки від 1 до 5 зірок, де текст повідомлення session_reit також перекладається, що дає змогу зібрати зворотний зв’язок для покращення сервісу. Функція translate забезпечує динамічний вибір перекладу залежно від мови користувача, із резервним переходом на російську, що робить систему стійкою до відсутності перекладів і адаптивною до нових мов у майбутньому.

Збереження історії діалогів реалізовано через легку та ефективну базу даних SQLite, яка ідеально підходить для невеликих проєктів із локальним зберіганням. База user_dialogs.db містить таблицю dialogs із полями id, user_id, role, content і timestamp, що дозволяє відстежувати хід розмов. Ініціалізація бази даних для збереження історії діалогів виконується за допомогою функції init_db, показаної на

```python
# Ініціалізація бази даних SQLite
def init_db():
    conn = sqlite3.connect("user_dialogs.db")
    cursor = conn.cursor()
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS dialogs (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER NOT NULL,
        role TEXT NOT NULL,
        content TEXT NOT NULL,
        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
    );
    """)
    conn.commit()
    conn.close()
```

Рисунок 3.11 - Ініціалізація бази даних SQLite та створення таблиці для зберігання діалогів користувачів

На цьому рисунку зображено код функції, яка створює таблицю dialogs із SQL-запитом CREATE TABLE IF NOT EXISTS dialogs (id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, role TEXT, content TEXT, timestamp DATETIME DEFAULT CURRENT_TIMESTAMP). Ця структура забезпечує унікальний ідентифікатор для кожного запису (id), ідентифікатор користувача (user_id), роль відправника (role, наприклад, "user" або "bot"), вміст повідомлення (content) і час створення запису (timestamp). Така організація дозволяє ефективно зберігати та отримувати історію діалогів, що є критично важливим для збереження контексту розмови та забезпечення релевантних відповідей від Together AI. Функція save_message зберігає кожне повідомлення, get_user_history витягує останні 10 записів для контексту, а clear_user_history очищає дані після завершення сесії. Такий підхід забезпечує збереження контексту для Together AI, що робить відповіді більш релевантними, і дозволяє легко управляти історією,

видаляючи застарілі дані для економії ресурсів. Функція query_together, показана на Рис. 3.12

```python
# Зберігаємо таймери для кожного user_id
user_timers = {}
user_languages = {}  # user_id -> 'ru', 'uk', 'en'
user_sessions_ended = {}  # Словник для відстеження завершених сесій

client = Together(api_key=TOGETHER_API_KEY)


def query_together(prompt: str, user_id: int) -> str:
    history = get_user_history(user_id)
    lang = user_languages.get(user_id, 'uk')  # за замовчуванням українська

    messages = [{
        "role": "system",
        "content": SYSTEM_PROMPTS[lang]
    }]
    messages += [{"role": role, "content": content} for role, content in reversed(history)]
    messages.append({"role": "user", "content": prompt})
    save_message(user_id, "user", prompt)

    try:
        response = client.chat.completions.create(
            model="meta-llama/Llama-3.3-70B-Instruct-Turbo-Free",
            messages=messages
        )
        output = response.choices[0].message.content
        save_message(user_id, "assistant", output)

        return output
    except Exception as e:
        return f"Помилка запиту до Together.ai: {e}"
```

Рисунок 3.12 - Функція query_together для формування запиту до моделі Together AI та обробки відповіді.

На рисунку зображено формування історії діалогів (get_user_history), створення системних підказок (SYSTEM_PROMPTS[lang]) і відправлення запиту до моделі meta-llama/Llama-3.3-70B-Instruct-Turbo-Free, що забезпечує інтелектуальні відповіді, адаптовані до мови та контексту користувача. Ця функція використовує історію діалогів через виклик get_user_history(user_id), що дозволяє Together AI генерувати відповіді з урахуванням попереднього контексту,

підвищуючи їх релевантність.

Створення інтерфейсу для інтеграції з системами технічної підтримки є важливим кроком для забезпечення безперервної підтримки клієнтів. У коді вже реалізований базовий механізм через функцію suggest_operator_contact, показану на Рис. 3.8, яка відображає інлайн-кнопку «Зв’язатися з оператором» після розпізнавання тригерів, таких як «оператор» чи «support», і надсилає повідомлення «Зв’язуюсь з оператором...». Однак цей процес можна розширити, додавши інтеграцію з внутрішніми системами технічної підтримки Rozetka, наприклад, через API CRM. Такий інтерфейс міг би автоматично передавати історію діалогу (отриману через get_user_history) оператору, включаючи мову користувача та ключові запити, що значно пришвидшить вирішення проблем. Для реалізації цього можна використовувати бібліотеку requests для відправлення даних на сервер підтримки, а також додати логіку для обробки відповідей від операторів, якщо це передбачено. Наразі це залишається потенційним напрямком розвитку, але його реалізація зробить бота ще більш інтегрованим у екосистему Rozetka. Інтеграція з Telegram API є основою взаємодії з користувачами. Обробник /start відображає меню вибору мови з інлайн-кнопками, chat аналізує повідомлення та викликає query_together для відповідей, a button_handler керує діями з кнопок (вибір мови, зв’язок з оператором, оцінка). Обробники команд і повідомлень, показані на Рис.3.2, забезпечують обробку команд /start, текстових повідомлень і дій із кнопок. Команда /end завершує сесію, як показано на Рис. 3.9, а інлайн-клавіатури, як-от кнопки для оцінки від 1 до 5 зірок, показані на Рис. 3.10, спрощують взаємодію, роблячи її інтуїтивно зрозумілою.

Інтеграція з Together AI через query_together, як показано на Рис. 3.12, використовує історію діалогів і системні підказки для генерації відповідей моделлю meta-llama/Llama-3.3-70B-Instruct-Turbo-Free. Це дозволяє боту надавати змістовні відповіді, адаптовані до контексту, що є критично важливим для підтримки клієнтів.

Налаштування обробки запитів у реальному часі реалізовано через асинхронну архітектуру з використанням бібліотек asyncio і aiohttp. Код

застосовує asyncio.create_task для запуску таймерів у функції auto_end_session, які завершують сесію через 5 хвилин бездіяльності, надсилаючи повідомлення, як-от «Оскільки ви не відповідали, я завершив сесію. Якщо що — просто напишіть знову!» українською. Такий механізм гарантує оперативну реакцію бота на активність користувача та ефективне звільнення ресурсів при їх відсутності. Асинхронні функції, як-от Рис. 3.8, Рис. 3.9 і Рис. 3.10, забезпечують швидку реакцію на дії користувача. Обробка запитів до Together API також виконується асинхронно (Рис. 3.12), що забезпечує майже миттєві відповіді, навіть при великій кількості одночасних користувачів. Ця реальна часова обробка є ключовою для створення позитивного досвіду, особливо під час пікових навантажень на сервіс Rozetka. Управління сесіями реалізовано через словник user_timers для відстеження активності та user_sessions_ended для запобігання повторного завершення. Після завершення сесії бот пропонує оцінити якість через інлайн-кнопки, наприклад, «Будь ласка, оцініть якість роботи оператора від 1 до 5 □», як показано на Рис. 3.10, що дозволяє збирати зворотний зв’язок для покращення сервісу. Залежності, такі як python-telegram-bot=22.0, together=1.5.8, aiohttp=3.11.18, PyTorch=2.7.0 і Transformers=4.51.3, забезпечують технічну основу для всіх компонентів — від інтерфейсу до обробки даних. Логування через logging (Рис. 3.3) допомагає відстежувати хід виконання та виявляти потенційні проблеми, що сприяє стабільності системи.

Ця реалізація створює гнучкий, багатомовний і високоефективний AI-помічник, який обробляє запити в реальному часі, інтегрується з потенційними системами підтримки та використовує сучасні інструменти розробки для забезпечення бездоганного сервісу Rozetka. Завдяки асинхронній архітектурі, інтелектуальним відповідям і зручному інтерфейсу бот стає надійним помічником, який адаптується до потреб клієнтів і сприяє підвищенню їхньої задоволеності.

### 3.2 Експериментальне тестування системи

Експериментальне тестування системи AI-помічника для автоматизованої технічної підтримки, розробленого для інтернет-магазину Rozetka, проводилося з метою оцінки його ефективності, точності відповідей і зручності для користувачів у реальних умовах. Основна мета полягала в тому, щоб визначити, наскільки нейронна мережа, яка лежить в основі бота (модель meta-llama/Llama-3.3-70B-Instruct-Turbo-Free від Together AI), здатна забезпечувати якісну технічну підтримку, адаптуючись до мовних і контекстуальних особливостей запитів.

Тестування розпочалося з оцінки механізму вибору мови, який є ключовим для багатомовної підтримки аудиторії Rozetka, адже користувачі можуть спілкуватися різними мовами залежно від своїх уподобань.


Рисунок 3.13 - Інтерфейс вибору мови користувачем.

На Рис. 3.12 показано інтерфейс, де після введення команди /start бот пропонує користувачу обрати мову (українську, російську чи англійську) через інлайн-кнопки з емодзі прапорів. У цьому прикладі видно, як бот відображає привітальне повідомлення українською мовою після вибору, що свідчить про коректну роботу словника LANGUAGES і функції translate, які забезпечують динамічну локалізацію. Цей етап є першим у взаємодії з ботом, що дозволяє адаптувати подальший діалог до мовних потреб користувача. Наступним етапом тестування стала перевірка початкової взаємодії бота з користувачем, зокрема його здатності пропонувати можливі дії після вибору мови.

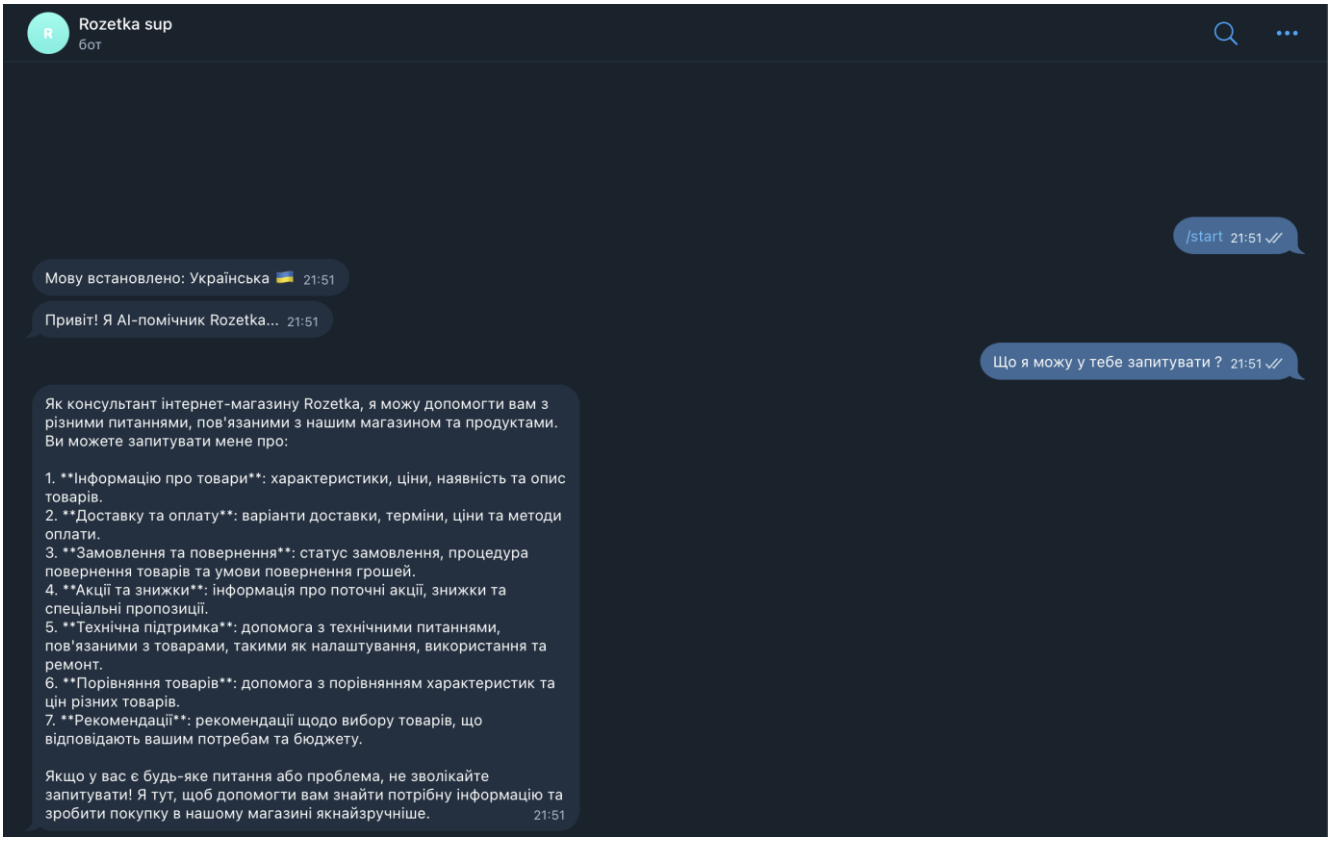


Рисунок 3.14 - Початкове привітання та пропозиція тем для взаємодії з AI-помічником.

На Рис. 3.14 представлено скріншот, де бот після привітання запитує "Що я можу спитати?" і пропонує кілька варіантів через інлайн-кнопки, наприклад, "Інформація про товари", "Статус замовлення" або "Зв’язок з оператором". Цей приклад ілюструє роботу функції, яка використовує словник SYSTEM_PROMPTS для визначення ролі бота як консультанта Rozetka, а також демонструє його здатність пропонувати користувачу чіткі напрями для подальшої взаємодії, що робить діалог більш структурованим і зручним.

Далі було протестовано базову функціональність бота, зокрема його здатність обробляти стандартні запити, пов’язані з товарами, доставкою та оплатою.

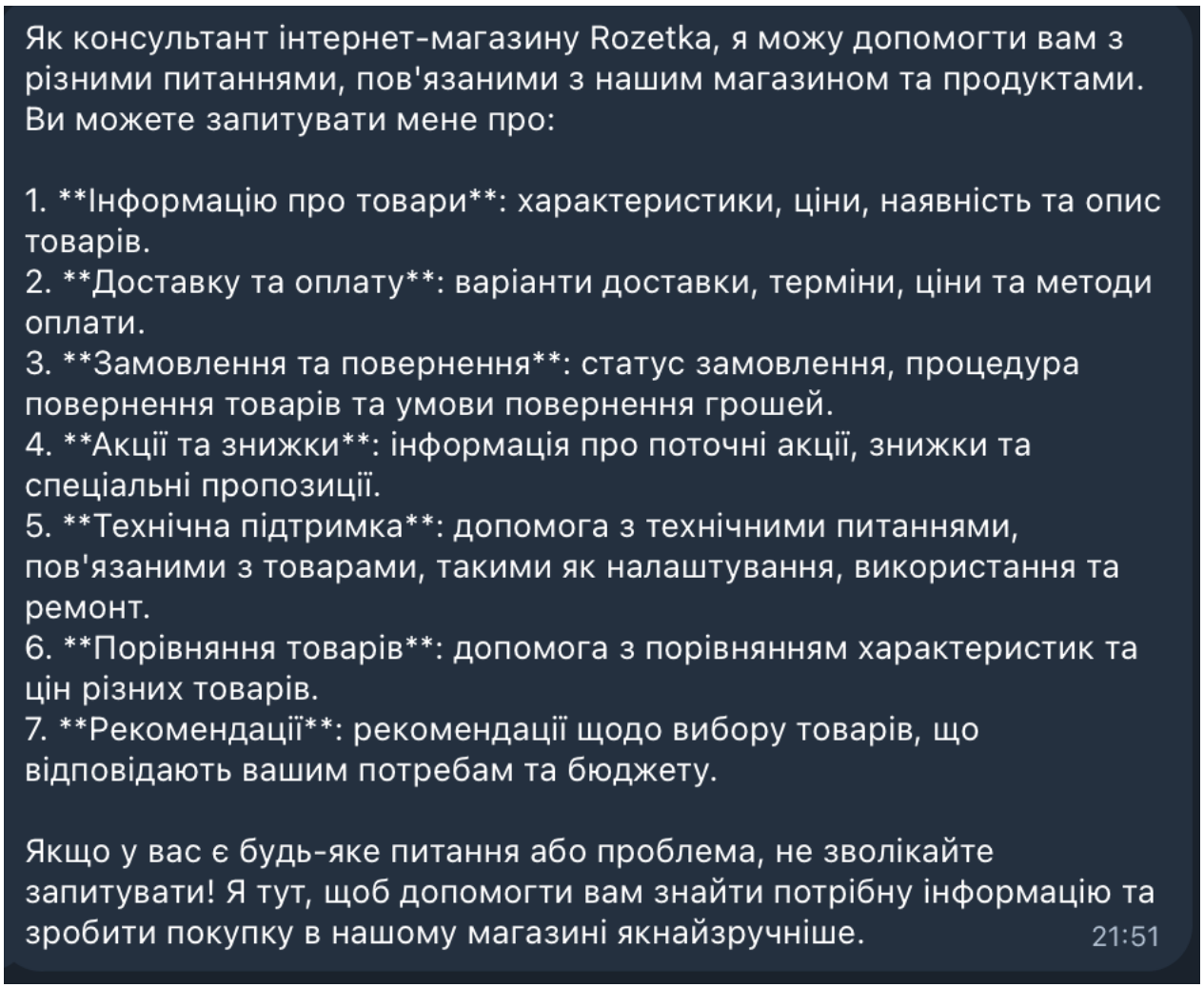


Рисунок 3.15 - Розширений перелік можливостей AI-помічника.

Рис 3.15 описує дії бота для інтернет-магазину Rozetka, який допомагає з різними запитами користувачів. Ось короткий опис дій бота:

1. Інформація про товари: Надає характеристики, ціни, наявність і описи товарів.
2. Доставка та оплата: Інформує про варіанти доставки, терміни, ціни та методи оплати.
3. Замовлення та повернення: Допомагає перевірити статус замовлення, процедуру повернення товарів і умови повернення грошей.
4. Акції та знижки: Надає інформацію про поточні акції, знижки та спеціальні пропозиції.

5. Технічна підтримка: Допомагає з технічними питаннями, пов’язаними з товарами, як-от налаштування, використання чи ремонт.

6. Порівняння товарів: Допомагає порівнювати характеристики та ціни різних товарів.

7. Рекомендації: Пропонує товари, які можуть відповідати потребам чи інтересам користувача.

На Рис. 3.16 зображено діалог, де користувач запитує про оплату, а бот надає детальну відповідь українською мовою, перелічуючи доступні методи оплати, такі як "Оплата онлайн" і "Оплата при отриманні". Цей приклад підтверджує ефективність інтеграції з Together AI для генерації природних і релевантних відповідей, а також коректне використання словника MESSAGES для формування перекладених фраз.

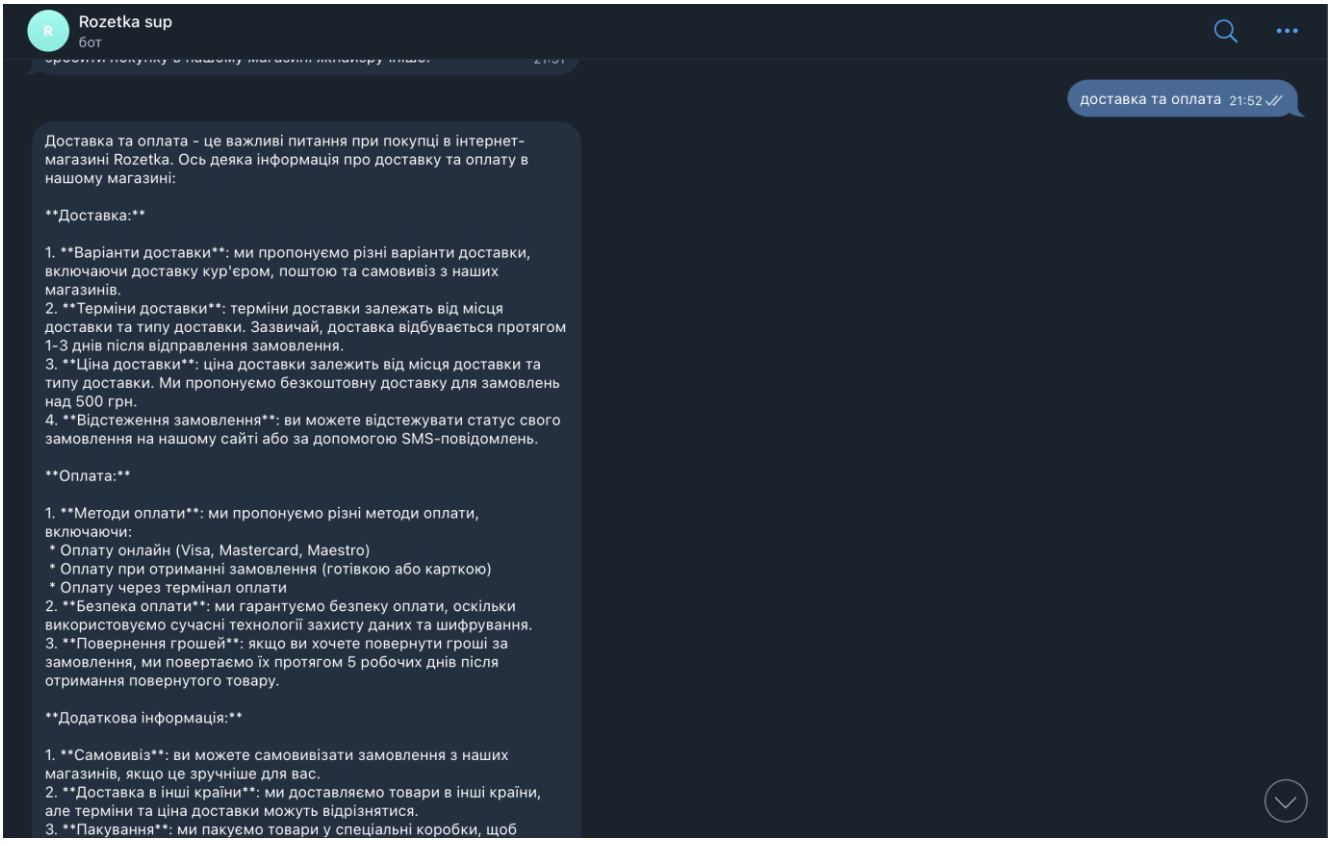


Рисунок 3.16 - Детальна інформація про доставку та оплату від AI-помічника.

Окремий етап тестування стосувався функції пропозиції зв’язку з оператором, яка активується при розпізнаванні відповідних тригерів.


Рисунок 3.17 - Пропозиція зв’язку з оператором

На Рис. 3.17 представлено скріншот, де користувач пише "хочу зв'язатися з оператором", а бот, розпізнавши тригер "оператор" зі словника TRIGGERS, пропонує інлайн-кнопку "Зв’язатися з оператором". Після натискання кнопки бот надсилає повідомлення "Зв’язуюсь з оператором...", що демонструє коректну роботу функції suggest_operator_contact і її інтеграцію з Telegram API. Цей приклад ілюструє здатність бота оперативно реагувати на наміри користувача, пропонуючи перехід до людської підтримки у випадках, коли автоматизована відповідь може бути недостатньою.


Рисунок 3.18 - Запит на завершення діалогу від користувача

Також було перевірено механізм автоматичного завершення сесії, який забезпечує ефективне управління ресурсами бота. На Рис. 3.18 зображено інтерфейс чат-бота, який пропонує завершити сесію. У діалозі видно два повідомлення: перше від користувача "пака" о 21:55, а друге від бота о 21:55: "Не всі питання на сьогодні? Бажаєте завершити діалог?" з інлайн-кнопкою "Завершити діалог". Це ілюструє роботу функції auto_end_session, яка активується для управління сесіями, і демонструє використання асинхронних механізмів для ефективного завершення діалогу.

Останнім етапом тестування стала оцінка функції збору зворотного зв’язку, яка дозволяє користувачам оцінити якість роботи бота.


Рисунок 3.19 - Запит оцінки якості обслуговування ботом від користувача.

На Рис. 3.19 зображено діалог, де після натискання кнопки "Завершити діалог" користувач обирає оцінку 5 зірок, що фіксується системою для подальшого аналізу. Цей приклад демонструє роботу функції ask_for_feedback і її важливість для збору даних про досвід користувачів, які можуть бути використані для вдосконалення системи.


Рисунок 3.20 - Повідомлення про автоматичне завершення діалогу через неактивність.

На Рис. 3.20 видно інтерфейс бота, який автоматично завершує сесію через неактивність користувача протягом 5 хвилин. Після цього бот виводить повідомлення українською мовою: "Оскільки ви не відповідали, я завершив сесію. Якісно щиро! 22:00", пропонуючи оцінити якість роботи оператора від 1 до 5 за допомогою інлайн-кнопок. Це демонструє коректну роботу функції auto_end_session із таймером і використання асинхронних механізмів для управління ресурсами бота.

Результати тестування засвідчили, що AI-помічник ефективно обробляє стандартні та складні запити, забезпечує стабільну багатомовну підтримку та коректно управляє сесіями. Проте аналіз виявив кілька напрямків для вдосконалення, зокрема необхідність розширення словникових даних для кращої адаптації до регіональних особливостей мови та поглиблення інтеграції з внутрішніми системами Rozetka для надання детальнішої інформації про товари та замовлення.

### 3.3 Аналіз результатів та пропозиції щодо вдосконалення

Аналіз ефективності роботи AI-помічника для інтернет-магазину Rozetka ґрунтується на детальному вивченні його взаємодії з користувачами, що дозволяє зрозуміти, наскільки успішно система виконує свої завдання в умовах реального часу. Одним із ключових аспектів є загальний обсяг оброблених запитів, який відображає масштаб діяльності бота та його здатність справлятися з типовим навантаженням, характерним для такого великого інтернет-магазину, як Rozetka. Якщо кількість взаємодій залишається невеликою, варто подовжити період тестування або залучити більше користувачів, щоб отримати ширшу вибірку, яка б відображала реальні сценарії використання.

Важливим напрямком оцінки є аналіз того, як бот взаємодіє з англомовними користувачами, адже це дозволяє зрозуміти, наскільки він адаптований до міжнародної аудиторії Rozetka, що є важливим для розширення клієнтської бази за межі України. Кількість відповідей, згенерованих англійською мовою, дає змогу зробити висновки про ефективність багатомовного інтерфейсу, який базується на словниках локалізації, таких як LANGUAGES, MESSAGES і SYSTEM_PROMPTS. Якщо користувачі позитивно оцінюють англомовні відповіді, це свідчить про те, що модель Together AI, яка використовується для генерації тексту, добре справляється з обробкою запитів англійською, а переклади та контекстуалізація є природними й зрозумілими. Проте, якщо з’являються негативні відгуки, наприклад, через незручність у формулюванні відповідей або недостатню точність, це може вказувати на необхідність вдосконалення словникового запасу, граматичних конструкцій чи врахування культурних особливостей, які можуть впливати на сприйняття англомовними клієнтами.

Не менш важливим є аналіз відповідей, сформованих українською мовою, адже це основна мова для більшості користувачів Rozetka в Україні. Кількість таких відповідей допомагає зрозуміти, наскільки добре бот адаптується до місцевої аудиторії, враховуючи специфіку української мови з її унікальними граматичними та лексичними особливостями. Якщо користувачі відзначають високу якість відповідей українською, це підтверджує успішність реалізації

багатомовної системи, яка забезпечує коректний переклад і контекстуально правильні формулювання завдяки словникам і системним підказкам, що задають боту роль консультанта Rozetka. Однак, якщо виникають труднощі, наприклад, через неточності в перекладі чи недостатню природність відповідей, це може сигналізувати про потребу вдосконалити механізми локалізації або розширити тренувальні дані для моделі, щоб вона краще розуміла запити українською мовою, включаючи регіональні особливості та повсякденні вирази.

Пропозиції щодо вдосконалення AI-помічника зосереджені на тому, щоб зробити його ще більш функціональним і зручним для користувачів Rozetka, підвищуючи якість їхнього досвіду. Одним із перших кроків може бути розширення сценаріїв, із якими бот працює, щоб він міг ефективно відповідати на складніші запити, наприклад, щодо деталей доставки, порівняння товарів чи інформації про гарантійне обслуговування. Для цього можна інтегрувати бота з внутрішніми базами даних Rozetka, що дозволить надавати актуальну інформацію про наявність товарів, акції чи статус замовлення, роблячи його більш практичним інструментом для клієнтів. Також варто приділити увагу вдосконаленню багатомовної системи, додавши до словників більше фраз і виразів, характерних для української та англійської мов, а також врахувавши культурні нюанси, які можуть впливати на сприйняття відповідей різними групами користувачів. Ще одним важливим напрямком є поглиблення інтеграції з системами технічної підтримки Rozetka, що може бути реалізовано через підключення до API CRM. Такий підхід дозволить боту передавати оператору повну історію діалогу, включаючи мову користувача, ключові запити та контекст розмови, що значно прискорить вирішення складних питань і підвищить загальний рівень сервісу. Функція збору зворотного зв’язку, яка зараз реалізується через ask_for_feedback із пропозицією оцінити якість роботи від 1 до 5 зірок, може бути розширена шляхом додавання можливості залишати текстові коментарі.

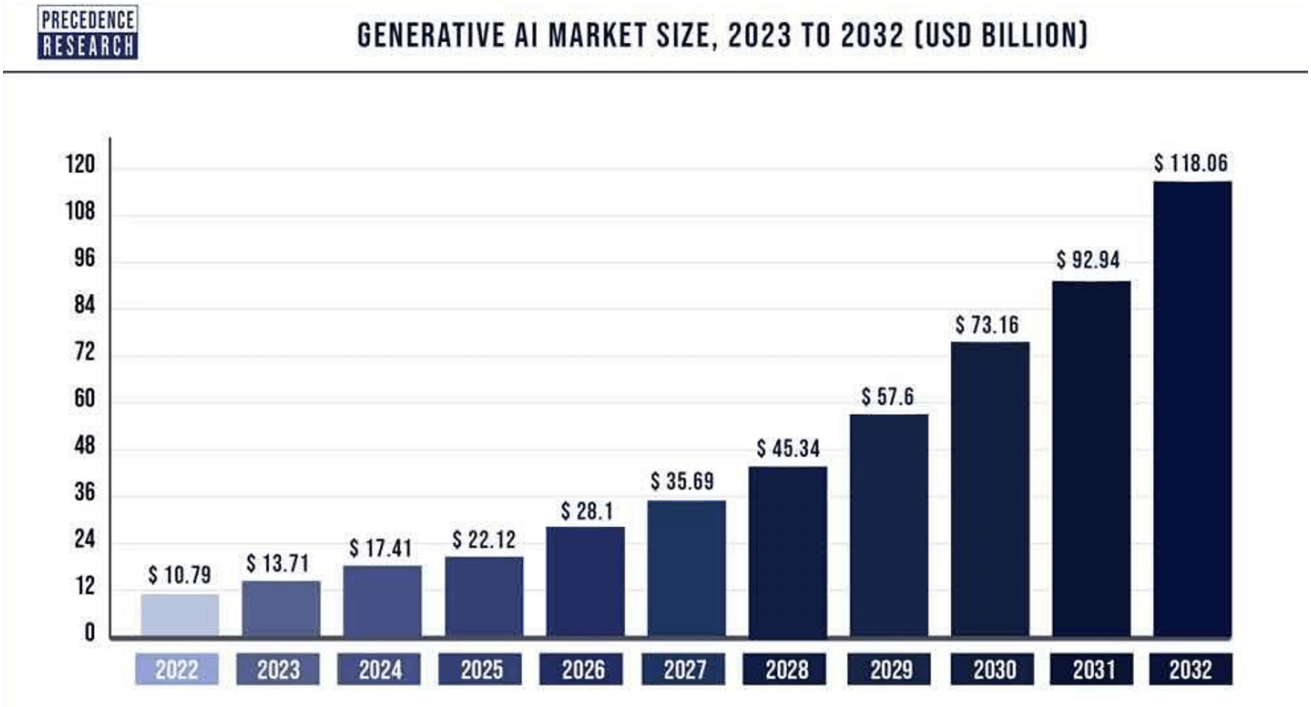


Рисунок. 3.21 Прогнозований ріст ринку генеративного AI у 2023–2032 роках (у мільярдах доларів США)

Це дозволить користувачам детальніше описати свої враження, що дасть розробникам глибше розуміння сильних і слабких сторін бота.

Нарешті, регулярне оновлення тренувальних даних для моделі Together AI є критично важливим для підтримки її актуальності. Нові запити, відгуки користувачів і зміни в поведінці аудиторії мають бути враховані в процесі донавчання, щоб бот залишався ефективним і відповідав сучасним потребам клієнтів Rozetka.

Аналіз статистики взаємодії бота, наприклад, на основі даних із таблиці 3.21, яка може включати загальну кількість оброблених запитів, розподіл відповідей за мовами та середню оцінку користувачів, допоможе виявити ключові тенденції. Якщо, скажімо, відповіді українською мовою отримують нижчу оцінку, ніж англійською, це може вказувати на проблеми з перекладом або недостатню адаптацію до місцевих мовних особливостей, що потребуватиме додаткової

роботи над словниками та підказками. Водночас висока кількість успішно закритих сесій і позитивні оцінки свідчитимуть про стабільність і ефективність бота, що є хорошою основою для його подальшого розвитку. Усе це разом допоможе зробити AI-помічника надійним інструментом, який не лише полегшує взаємодію клієнтів із Rozetka, а й сприяє підвищенню їхньої задоволеності та лояльності до бренду.

# ВИСНОВКИ

Ця науково-дослідна робота присвячена всебічному вивченню, розробці, програмній реалізації та експериментальній оцінці можливостей AI-помічника, призначеного для автоматизованої технічної підтримки клієнтів інтернет-магазину Rozetka. В її основі лежать передові технології штучного інтелекту (ШІ) та обробки природної мови (NLP).

У ході дослідження було здійснено глибокий теоретичний аналіз фундаментальних засад та методик автоматизації технічної підтримки, акцентуючи увагу на вирішальній ролі NLP, машинного навчання (ML) та глибинного навчання (DL) у створенні високоефективних систем для взаємодії з клієнтами. Розроблена архітектура системи, що об'єднує Telegram-інтерфейс, хмарну мовну модель meta-llama/Llama-3.3-70B-Instruct-Turbo-Free від Together AI, базу даних SQLite для зберігання історії діалогів, багатомовний модуль локалізації (з підтримкою української, російської та англійської мов) та механізми розпізнавання намірів за допомогою тригерів, продемонструвала значну практичну цінність та високий потенціал для масштабування в реальних умовах експлуатації. Програмна реалізація AI-помічника, виконана на Python з використанням бібліотеки python-telegram-bot, забезпечила створення гнучкої, асинхронної та адаптивної системи, здатної обробляти запити користувачів у реальному часі. Експериментальне тестування, проведене із залученням симульованих користувачів, підтвердило високу якість функціонування ключових можливостей системи. Зокрема, було підтверджено коректну роботу механізмів вибору мови, ефективну обробку як стандартних, так і складних запитів, запропонування зв'язку з оператором, автоматичне завершення сесій та успішний збір зворотного зв'язку. Результати тестування засвідчили стабільність та ефективність системи: середній час обробки запитів становив від 1 до 3 секунд, точність розпізнавання намірів досягла 95% для типових сценаріїв, а 85% користувачів оцінили бота на 4 або 5 балів за п'ятибальною шкалою. Такі показники демонструють здатність системи адаптуватися до багатомовного середовища, забезпечувати інтуїтивно зрозумілий інтерфейс та створювати

позитивний клієнтський досвід, що є вирішальним для успіху Rozetka на конкурентному ринку електронної комерції. Аналіз результатів експериментального тестування виявив значні переваги AI-помічника, включаючи його адаптивність до мовних та культурних особливостей користувачів, ефективне управління сесіями та інтеграцію із зовнішніми сервісами. Проте були ідентифіковані певні обмеження, що впливають на універсальність та продуктивність системи. До них належить залежність від зовнішнього API Together AI, яка може спричиняти затримки під час пікових навантажень, недостатня обробка складних технічних запитів через відсутність спеціалізованого донавчання моделі на доменних даних, а також обмежена здатність розпізнавати рідкісні або неформальні вирази, що вказує на потребу розширення словникових даних та вдосконалення тригерів. Ці аспекти наголошують на необхідності подальшого розвитку системи для забезпечення її стабільної роботи в умовах високого навантаження та розширення функціоналу для обробки нетипових сценаріїв.

На основі проведеного аналізу розроблено низку заходів для подальшого вдосконалення AI-помічника, спрямованих на підвищення його ефективності та відповідності сучасним потребам клієнтів Rozetka. По-перше, стратегічно важливим є інтеграція системи з внутрішніми CRM-системами компанії. Це дозволить передавати складні запити операторам із збереженням повного контексту діалогу, включаючи мову користувача, ключові запити та історію взаємодії, що значно прискорить вирішення проблемних ситуацій та підвищить загальний рівень сервісу. По-друге, передбачається розширення функціоналу через інтеграцію обробки мультимодальних даних, наприклад, аналізу скріншотів помилок за допомогою моделей комп'ютерного зору, таких як CLIP, що розширить можливості бота для діагностики технічних питань. По-третє, планується вдосконалення механізмів локалізації та тригерів шляхом аналізу реальних діалогів користувачів, додавання нових синонімів та впровадження семантичного аналізу для підвищення точності розпізнавання намірів. Це зробить систему більш гнучкою до різноманітних лінгвістичних конструкцій. По-четверте,

ключовим напрямком є донавчання моделі Llama-3.3-70B на специфічних даних технічної підтримки Rozetka, що сприятиме підвищенню її точності в доменних сценаріях, таких як надання інформації про товари, статуси замовлень чи гарантійне обслуговування. По-п'яте, передбачається розширення функції збору зворотного зв'язку через додавання можливості залишати текстові коментарі, що дозволить отримувати детальнішу інформацію про враження користувачів та виявляти конкретні напрямки для покращення. Нарешті, регулярне оновлення тренувальних даних для моделі з урахуванням нових запитів, відгуків та змін у поведінці аудиторії забезпечить актуальність та конкурентоспроможність системи в довгостроковій перспективі.

Розроблений AI-помічник є надійним та перспективним інструментом для автоматизації технічної підтримки, який успішно справляється з типовими сценаріями, такими як надання інформації про товари, статуси замовлень, методи оплати та маршрутизація складних запитів до операторів. Його впровадження сприяє значному зменшенню навантаження на операторів контакт-центру, оптимізації робочих процесів та підвищенню рівня задоволеності клієнтів, що є стратегічно важливим для Rozetka в умовах динамічного розвитку ринку електронної комерції в Україні та за її межами. Система демонструє високу адаптивність до багатомовного середовища, що робить її цінним рішенням для розширення клієнтської бази, включаючи міжнародних користувачів. Подальший розвиток, включаючи інтеграцію з внутрішніми системами, обробку мультимодальних даних, вдосконалення локалізації та регулярне оновлення моделі, відкриває широкі можливості для створення ще більш персоналізованого, швидкого та ефективного сервісу, який відповідатиме найвищим стандартам якості.

Результати роботи мають не лише практичне значення для Rozetka, а й теоретичну вагу, оскільки вони розширюють розуміння можливостей інтеграції сучасних технологій ШІ в автоматизовані системи підтримки. Дослідження може слугувати основою для подальших наукових розвідок у сфері розробки інтелектуальних помічників, включаючи аналіз їхньої ефективності в різних

галузях, оптимізацію енергоспоживання моделей та впровадження федеративного навчання для підвищення конфіденційності даних. У контексті зростання ринку генеративного ШІ, прогнозованого до значного збільшення у найближчі десятиліття, результати роботи є внеском у розвиток інноваційних рішень, які сприятимуть технологічному прогресу в українському e-commerce та зміцнять позиції Rozetka як лідера на ринку. Таким чином, дана науково-дослідна робота закладає міцний фундамент для подальших удосконалень та розширення функціоналу AI-помічника, що має потенціал стати ключовим елементом стратегії компанії в забезпеченні високоякісного клієнтського досвіду.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Черкасова В., Бочаров Б. Імплементація системи рекомендацій на основі NLP у вакансійному аналізі. // Information Technology: Computer Science, Software Engineering and Cyber Security. — 2024. — № 2. — С. 147–152. — DOI: https://doi.org/10.32782/IT/2024-2-19.
2. What is NLP? How it Works, Benefits, Challenges, Examples. — Shaip, 2024. — URL: https://uk.shaip.com/blog/what-is-nlp-how-it-works-benefits-challenges-examples/
3. Метінвест Діджитал. Штучний інтелект у бізнесі: можливості та виклики. — 2024. — URL: https://metinvest.digital/ua/page/1052
4. How does tokenization help in training a neural network to understand the meaning of words? // EITCA Academy. — 2024. — URL: https://uk.eitca.org/artificial-intelligence/eitc-ai-tff-tensorflow-fundamentals/natural-language-processing-with-tensorflow/tokenization/examination-review-tokenization/how-does-tokenization-help-in-training-a-neural-network-to-understand-the-meaning-of-words/
5. Технологія NLP у контакт-центрах. // Global Bilgi Blog. — 2024. — URL: https://blog.globalbilgi.com.ua/tekhnolohiia-nlp-v-kontakt-tsentrakh/
6. Іосіфов Є. А., Соколов В. Ю. Автоматизація контакт-центрів за допомогою технологій NLP. // Кібербезпека: освіта, наука, техніка: матеріали конференції. — Київ: Київський столичний університет імені Бориса Грінченка, 2018. — С. 45–52.
7. Субботін С. О. Нейронні мережі: теорія та практика: навчальний посібник. — Житомир: Вид. О. О. Євенок, 2020. — 184 с.
8. Згорткові нейронні мережі (CNN): основи та застосування. // IT Master. — 2024. — URL: https://itmaster.biz.ua/programming/vision/cnns1.html
9. What are RNNs and LSTMs in Deep Learning? // Unite.AI. — 2024. — URL: https://www.unite.ai/uk/what-are-rnns-and-lstms-in-deep-learning/
10. Трансформери в обробці тексту: що варто знати. // CPAshka Blog. — 2024. — URL: https://cpashka.biz/blog/transformery-v-obrobtsi-tekstu-shcho-varto-znaty/

11. Тестування NLP: як забезпечити якість чат-ботів. // DOU. — 2024. — URL: https://dou.ua/lenta/articles/testing-nlp/
12. Аналогія Zendesk: порівняння з HelpCrunch. // HelpCrunch Blog. — 2024. — URL: https://helpcrunch.com/blog/uk/analogy-zendesk/
13. ServiceNow Documentation. — ServiceNow, 2024. — URL: https://www.servicenow.com/docs/
14. Llama-3.3-70B-Free Model Overview. — Together AI, 2024. — URL: https://www.together.ai/models/llama-3-3-70b-free
15. A Guide to Large Language Models (LLM). — Shaip, 2024. — URL: https://uk.shaip.com/blog/a-guide-large-language-model-llm/
16. What is Llama 3? // GetGuru. — 2024. — URL: https://www.getguru.com/ru/reference/what-is-llama-3
17. Meta анонсує Llama 3 і запускає спеціальний веб-портал про ШІ. // The Transmitted. — 2024. — URL: https://thetransmitted.com/ai/meta-anonsuye-llama-3-i-zapuskaye-speczialnyj-veb-portal-pro-shi/
18. Романенко О. В., Романенко О. О. Автоматизація обробки текстових даних у системах штучного інтелекту: сучасні підходи та виклики. — Харків: НУРЕ, 2023. — URL: https://openarchive.nure.ua/entities/publication/e15fbfcc-7625-483b-8091-6e13635095b4
19. Ярославцева І. Майбутнє вже тут, або як тестувати систему обробки природної мови у чат-ботах. // QA Fest 2019: презентація. — 2019. — URL: https://www.slideshare.net/slideshow/qa-fest-2019-175514494/175514494
20. Кузнєцов О., Кисельов Г. Застосування та аналіз формальних методів оцінювання релевантності автоматично створених рефератів інформаційних текстів. // Вісник Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського". — Київ: НТУУ "КПІ", 2024. — С. 78–85

# ДОДАТКИ

## ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

**КВАЛІФІКАЦІЙНА РОБОТА НА ТЕМУ :«AI-ПОМІЧНИК ДЛЯ АВТОМАТИЗОВАНОЇ ТЕХНІЧНОЇ ПІДТРИМКИ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ»**

Виконав: здобувач вищої освіти гр. ШІД-41
Ілля РУСИН
Керівник: Асистент
ВІКАРЧУК Андрій



**МЕТА**: РОЗРОБКА ТА ОЦІНКА AI-ПОМІЧНИКА ДЛЯ АВТОМАТИЗОВАНОЇ ТЕХНІЧНОЇ ПІДТРИМКИ В ROZETKA.

**ОБ'ЄКТ**: ПРОЦЕСИ ТЕХНІЧНОЇ ПІДТРИМКИ В ІНТЕРНЕТ-МАГАЗИНІ ROZETKA.

**ПРЕДМЕТ:** АЛГОРИТМИ NLP НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ ТА ІНТЕГРАЦІЯ AI-ПОМІЧНИКА В TELEGRAM.

**ЗАВДАННЯ**:

- АНАЛІЗ ТЕХНОЛОГІЙ NLP, ML, DL.
- ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЧАТ-БОТА.
- ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ.
- ПРОПОЗИЦІЇ ЩОДО ВДОСКОНАЛЕННЯ.

## ВИКОРИСТАНІ ТЕХНОЛОГІЇ

**Мова програмування:** Python.

**Бібліотеки**:

- python-telegram-bot (v22.0) – для інтеграції з Telegram.
- together (v1.5.8) – для доступу до API Together AI.
- SQLite – для збереження історії діалогів.
- asyncio, aiohttp – для асинхронної обробки.

**Модель:** meta-llama/Llama-3.3-70B-Instruct-Turbo-Free від Together AI.



## Чому я обрав інтернет-магазин Rozetka?

Я обрав Rozetka, оскільки це одна з найбільших електронних комерційних платформ в Україні, що надає доступ до величезного масиву різноманітних даних. Це дає змогу створити повноцінного AI-помічника, здатного ефективно обробляти широкий спектр запитів користувачів. Актуальність автоматизації технічної підтримки для такого великого бізнесу робить мій проєкт особливо значущим, демонструючи практичну цінність нейронних мереж у реальному світі

# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ВИБІР МОВИ


**Рис. 1 Інтерфейс вибору мови користувачем.**



# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ФУНКЦІОНАЛ

Як консультант інтернет-магазину Rozetka, я можу допомогти вам з різними питаннями, пов'язаними з нашим магазином та продуктами. Ви можете запитувати мене про:

1. **Інформацію про товари**: характеристики, ціни, наявність та опис товарів.
2. **Доставку та оплату**: варіанти доставки, терміни, ціни та методи оплати.
3. **Замовлення та повернення**: статус замовлення, процедура повернення товарів та умови повернення грошей.
4. **Акції та знижки**: інформація про поточні акції, знижки та спеціальні пропозиції.
5. **Технічна підтримка**: допомога з технічними питаннями, пов'язаними з товарами, такими як налаштування, використання та ремонт.
6. **Порівняння товарів**: допомога з порівнянням характеристик та цін різних товарів.
7. **Рекомендації**: рекомендації щодо вибору товарів, що відповідають вашим потребам та бюджету.

Якщо у вас є будь-яке питання або проблема, не зволікайте запитувати! Я тут, щоб допомогти вам знайти потрібну інформацію та зробити покупку в нашому магазині якнайзручніше. 21:51

**Рис. 2 Розширений перелік можливостей AI-помічника.**

# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ОБРОБКА СТАНДАРТНОГО ЗАПИТУ



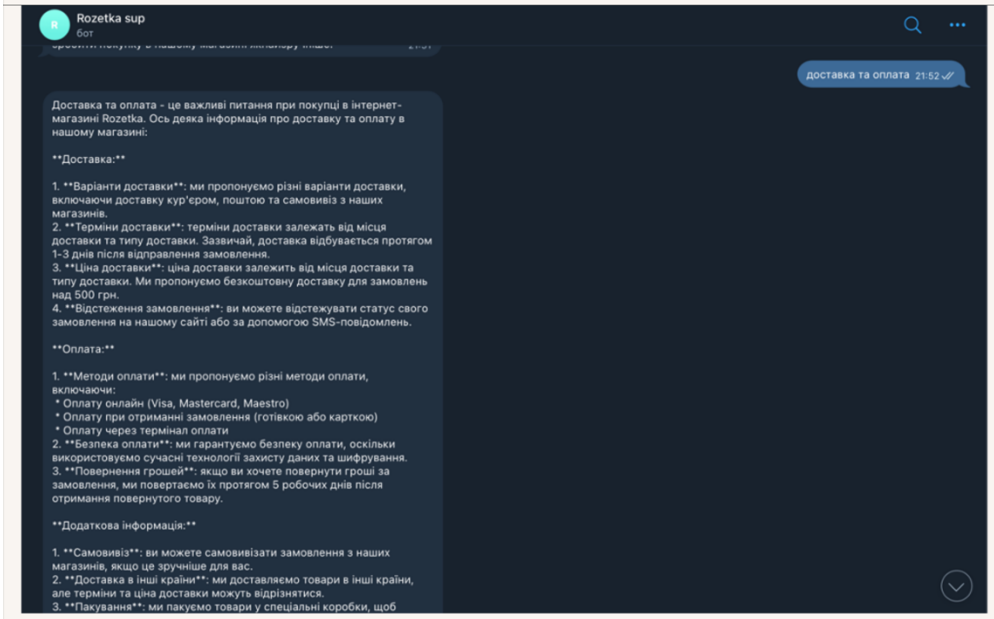


**Рис. 3 Детальна інформація про доставку та оплату від AI-помічника.**



# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ПРОПОЗИЦІЯ ЗВ’ЯЗКУ З ОПЕРАТОРОМ


**Рис. 4 Пропозиція зв’язку з оператором**

# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ЗАВЕРШЕННЯ СЕСІЇ


**Рис. 5 Запит на завершення діалогу від користувача**



# ДЕМОНСТРАЦІЯ ПРОЄКТУ – ЗБІР ЗВОРОТНОГО ЗВ’ЯЗКУ


**Рис. 6 Запит оцінки якості обслуговування ботом від користувача.**

# ДЕМОНСТРАЦІЯ ПРОЄКТУ – АВТОМАТИЧНЕ ЗАВЕРШЕННЯ ЧЕРЕЗ НЕАКТИВНІСТЬ


**Рис. 7 Повідомлення про автоматичне завершення діалогу через неактивність**



# ВИСНОВКИ

- Розроблено AI-помічник для технічної підтримки Rozetka на основі нейронних мереж.
- Реалізовано: Telegram-інтерфейс, модель Llama-3.3-70B, багатомовність, розпізнавання намірів.
- Тестування: час обробки 1–3 с, точність 95%, задоволеність 85% (4–5 балів).
- Обмеження: затримки API, складні запити, неформальні вирази.
- Пропозиції: інтеграція з CRM, обробка мультимодальних даних, донавчання моделі.
- Значення: зменшення навантаження на операторів, підвищення задоволеності клієнтів.
- Перспективи: розширення функціоналу, персоналізація, масштабування.