

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Створення чат боту з використанням NLP для
оптимізації ресурсів кол центру»**

Виконав студент групи ІІЗм-24-2
ШЕВЦОВ Ігор Володимирович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ШЕВЦОВ Ігор Володимирович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Створення чат боту з використанням NLP для оптимізації ресурсів кол центру»**,

затверджена Наказом закладу вищої освіти від **«18» вересня 2025 р. № 491/с**

Вихідні дані для роботи: **аналіз процесів взаємодії користувачів із системами підтримки, дослідження методів автоматизованої обробки текстових звернень та визначення намірів за допомогою нейронних мереж; вивчення підходів до побудови датасетів.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	22.09.2025р.	22.09.2025р.
2	Морозов А.В.	22.09.2025р.	22.09.2025р.
3	Морозов А.В.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена створенню інтелектуальної системи підтримки користувачів, що забезпечує автоматичне визначення намірів у текстових зверненнях на основі методів обробки природної мови та сучасних нейронних мереж. Пояснювальна записка складається зі вступу, трьох розділів, висновків та списку використаної літератури. У пояснювальній записці міститься 62 сторінки, 16 ілюстрацій, 1 таблиця, 4 додатки та 16 посилань на зовнішні джерела.

Метою роботи є розробка та експериментальне дослідження системи чат-боту, здатної автоматично класифікувати намір користувача, забезпечуючи тим самим оптимізацію роботи контакт-центрів та зменшення навантаження на операторів.

У роботі описано процес формування та підготовки україномовного доменного датасету звернень користувачів, розглянуто підходи до побудови моделей визначення намірів, включаючи базовий статистичний метод та трансформерну модель XLM-RoBERTa. Висвітлено етапи навчання моделей, проведено порівняльний аналіз їх точності та ефективності. Особливу увагу приділено інтеграції моделі у мікросервісну архітектуру, що включає Python-сервіс для інференсу, Node.js gateway для маршрутизації запитів та двосторонню комунікацію через WebSocket.

Представлено реалізацію веб-клієнта та адмін-панелі, що забезпечують взаємодію користувача з системою та моніторинг активних діалогів. Проведено тестування роботи чат-боту в різних сценаріях, що підтвердило коректність класифікації намірів та доцільність використання запропонованого підходу для автоматизації обробки звернень.

КЛЮЧОВІ СЛОВА: NLP, ВИЗНАЧЕННЯ НАМІРІВ, НЕЙРОННА МЕРЕЖА, ТРАНСФОРМЕР, ДАТАСЕТ, ЧАТ-БОТ, КЛАСИФІКАЦІЯ ТЕКСТУ.

ABSTRACT

The master's qualification thesis is devoted to the development of an intelligent user-support system capable of automatically identifying user intents in text messages using natural language processing methods and modern neural network architectures. The explanatory report consists of an introduction, three chapters, conclusions, and a list of references. The document contains 62 pages, 16 figures, 1 table, 4 appendices, and 16 references to external sources.

The aim of the work is the design and experimental evaluation of a chatbot system capable of automatically classifying user intents, thereby optimizing the operation of contact centers and reducing the workload on support agents.

The thesis describes the process of creating and preparing a Ukrainian-language domain-specific dataset of user messages, examines several approaches to intent classification, including a baseline statistical method and a transformer-based XLM-RoBERTa model. The stages of model training are detailed, and a comparative analysis of model performance and accuracy is provided. Special attention is given to integrating the trained model into a microservice architecture, which includes a Python inference service, a Node.js gateway for request routing, and bidirectional communication using WebSocket.

The implementation of a web-based client and an administrative panel is presented, enabling user interaction with the system and real-time monitoring of active dialogues. System testing under various scenarios confirmed the correctness of intent classification and demonstrated the practical value of the proposed solution for automating customer request processing.

KEYWORDS: NLP, INTENT CLASSIFICATION, NEURAL NETWORK, TRANSFORMER, DATASET, CHATBOT, TEXT CLASSIFICATION.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	9
1.1. Аналіз проблем і особливостей процесів сучасних кол-центрів	9
1.2. Аналіз наявних методів автоматизації кол-центрів	10
1.3. Постановка задачі дослідження.....	14
1.4. Вибір засобів реалізації.....	15
Висновки до розділу 1	17
РОЗДІЛ 2. МЕТОДИ РОЗВ’ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ НАМІРІВ КОРИСТУВАЧІВ	19
2.1. Методи побудови моделі визначення намірів	19
2.1.1. Створення датасету	21
2.1.2. Конфігурація та навчання моделі	23
2.2. Аналіз отриманих даних	26
2.3. Проектування додатку.....	29
Висновки до розділу 2.....	33
РОЗДІЛ 3. РЕАЛІЗАЦІЯ МІКРОСЕРВІСНОЇ ІНФРАСТРУКТУРИ ЧАТБОТУ З МОДУЛЕМ ВИЗНАЧЕННЯ ІНТЕНЦІЙ.....	36
3.1. Основні вимоги до системи	36
3.2. Вибір інструментальних засобів розробки.....	38
3.3. Реалізація окремих модулів чатботу.....	41
3.4. Опис інтерфейсу програмного продукту.....	43
Висновки до розділу 3	50
ВИСНОВКИ	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54
ДОДАТКИ	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

NLP – (англ. natural language processing) – обробка природної мови

AI – (англ. artificial intelligence) – штучний інтелект

ML – (англ. machine learning) – машинне навчання

IVR – (англ. interactive voice response) – інтерактивна голосова відповідь

ВСТУП

У сучасних умовах стрімкого розвитку цифрових сервісів та збільшення частки комунікацій, що здійснюються онлайн, бізнес дедалі частіше стикається з потребою швидко, точно та економічно обробляти великий обсяг клієнтських звернень. У сфері електронної комерції ця потреба особливо висока, оскільки користувачі очікують негайної реакції та оперативного вирішення питань, пов'язаних із доставкою, оплатою, товарами чи статусом замовлень. Традиційна модель роботи кол-центрів, що базується на участі операторів, не завжди здатна забезпечити необхідний рівень масштабованості, а зростання витрат на людські ресурси спонукає компанії до активнішого впровадження технологій автоматизації.

Одним із найбільш перспективних напрямів є застосування методів обробки природної мови, які забезпечують можливість машинного розуміння змісту текстових повідомлень. Сучасні трансформерні моделі дозволяють аналізувати мовні конструкції зі значно вищою точністю, ніж попередні генерації NLP-підходів, що робить їх придатними для розпізнавання намірів користувача навіть у разі варіативності мовлення, використання розмовних форм чи граматичних відхилень. У контексті української мови ця задача є додатково ускладненою через наявність суржику та широкого стилістичного різноманіття, що потребує формування якісних україномовних датасетів та адаптації моделей під реальні сценарії використання.

Упровадження інтелектуальних чат-ботів у службу підтримки створює можливість суттєво знизити навантаження на операторів та забезпечити цілодобовий доступ до сервісу без збільшення витрат на персонал. Однак ефективність таких рішень безпосередньо залежить від точності визначення намірів користувачів, оскільки саме цей етап визначає подальший сценарій діалогу. Тому розроблення й дослідження підходів до класифікації інтенцій українською мовою є важливим етапом розвитку локалізованих систем автоматизації клієнтської підтримки.

Метою роботи є розроблення підходу до автоматизованого визначення інтенцій користувачів у текстових зверненнях українською мовою з використанням сучасних моделей обробки природної мови з подальшою оцінкою придатності цього підходу для інтеграції в практичну систему чат-боту.

Завданнями роботи є аналіз сучасних підходів і методів обробки природної мови для задачі визначення намірів користувачів; формування спеціалізованого україномовного набору текстових даних у сфері електронної комерції; розроблення та навчання моделей машинного навчання для класифікації користувацьких намірів; а також аналіз можливостей практичного застосування отриманих результатів у системах чат-ботів клієнтської підтримки.

Об’єктом дослідження є процес автоматизованої обробки та інтерпретації текстових запитів у системах клієнтської підтримки.

Предметом дослідження є методи та моделі машинного навчання, що забезпечують класифікацію користувацьких намірів у діалогових системах, зокрема на основі трансформерних архітектур.

Наукова новизна роботи полягає у формуванні та використанні спеціалізованого україномовного набору даних для задачі визначення інтенцій у сфері електронної комерції, а також у дослідженні ефективності моделей трансформерного типу для класифікації реалістичних варіацій користувацьких висловлювань, включно з розмовними та змішаними мовними формами.

Практичне значення полягає у можливості застосування отриманих результатів для розроблення інтелектуальних чат-ботів, здатних автоматично інтерпретувати наміри клієнтів та знижувати навантаження на операторів кол-центру, що у свою чергу сприятиме підвищенню швидкості обробки звернень, покращенню користувацького досвіду та оптимізації бізнес-процесів підтримки.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз проблем і особливостей процесів сучасних кол-центрів

Сучасні кол-центри функціонують у середовищі, що характеризується високою динамічністю, збільшенням обсягів клієнтських звернень та розширенням спектра цифрових комунікацій. Дослідження, представлене у виданні *Applied System Innovation* [1], підкреслює, що компанії стикаються зі значними труднощами при обробці природної мови через варіативність формулювань, відсутність структурованості тексту та складність інтерпретації намірів користувачів. Зокрема, автори акцентують увагу на тому, що сучасні запити споживачів часто містять емоційні елементи, змішані стилі мовлення та неоднозначні конструкції, що ускладнює їх автоматизацію. Це створює передумови до збільшення навантаження на операторів і підвищення часу обробки звернень через необхідність ручного аналізу кожного випадку [2,3].

У контексті зростання обсягів звернень показовим є аналіз тенденцій розвитку індустрії кол-центрів, наведений у *Future of Call Centers: 30 Must-Know Statistics and Trends* [4]. У ньому відзначається, що з переходом значної частини сервісних процесів у цифрову площину кількість звернень систематично збільшується. Користувачі очікують швидких відповідей, а затримки в обслуговуванні стають одним із найбільш критичних чинників негативної оцінки сервісу. У тому ж джерелі наголошується, що тривале очікування відповіді оператором викликає у клієнтів невдоволення, а надмірне навантаження на кол-центри призводить до підвищення середнього часу обробки звернення. Таким чином, зростання кількості запитів без відповідної модернізації інструментів підтримки неминуче виявляє структурні обмеження традиційних моделей роботи кол-центрів.

Додатковий вимір проблематики розкривається у дослідженні О. Торреса та співавторів [5], яке фокусується на психологічному та організаційному аспектах роботи операторів. Автори доводять, що постійне навантаження та необхідність виконання великої кількості рутинних дій призводять до підвищеного рівня стресу, емоційного виснаження та зниження

продуктивності персоналу. Негативні наслідки цього явища проявляються у зниженні якості взаємодії з клієнтом, збільшенні кількості помилок та зростанні плинності кадрів. Висновки роботи підкреслюють: надмірна залежність від людського фактора в умовах інтенсивної роботи кол-центрів є однією з ключових перепон для масштабування сервісних процесів і забезпечення стабільності обслуговування [6].

Таким чином, аналіз наукових і практичних джерел демонструє комплексну природу проблем сучасних кол-центрів: лінгвістичну складність запитів, різке збільшення обсягів звернень та значне психологічне навантаження на персонал. Ці чинники взаємопов'язані та загалом формують середовище, у якому традиційні системи обробки звернень більше не здатні забезпечувати необхідну якість обслуговування. У таких умовах інтелектуальні системи аналізу природної мови набувають особливої актуальності, оскільки саме вони здатні виконувати попередню інтерпретацію запитів, автоматично визначати їхню інтенцію та забезпечувати масштабовану підтримку процесів клієнтського сервісу [7].

Впорядковане представлення проблеми логічно підводить до вибору теми дослідження, спрямованої на розроблення моделі класифікації намірів користувачів. Використання NLP-технологій у цьому контексті дозволяє не лише знизити навантаження на операторів, але й забезпечити високу швидкість реагування та стійкість сервісу навіть за умов значного зростання кількості звернень. Саме це обґрунтовує актуальність та практичну значимість розробки, описаної у даній роботі.

1.2. Аналіз наявних методів автоматизації кол-центрів

Автоматизація кол-центрів за останнє десятиліття пройшла шлях від простих телефонних сценаріїв до складних інтегрованих систем, здатних працювати з природною мовою, мультиканальними комунікаціями та даними клієнтів. Попри значний розвиток індустрії, жоден із наявних підходів не здатен повністю вирішити проблему швидкої та точної інтерпретації намірів

користувача, що залишається критичним елементом якісної підтримки. Аналіз реальних технологічних рішень дає змогу визначити наявні тенденції, сильні сторони та обмеження поточних систем [8].

Однією з найбільш розвинених груп технологій є омніканальні контакт-центри корпоративного рівня, такі як Genesys Cloud CX та Five9 Intelligent Cloud Contact Center [9,10]. Ці платформи забезпечують автоматизовану маршрутизацію звернень, управління чергами, збір аналітики та підтримку базових моделей обробки природної мови. Проте їхні вбудовані AI-модулі здебільшого орієнтовані на англomовні ринки, а можливості роботи з українською мовою залишаються обмеженими. Навіть попри використання алгоритмів ML для покращення маршрутизації, системи такого типу часто покладаються на шаблонні правила, що знижує їхню здатність адекватно інтерпретувати реальні розмовні формулювання користувачів. У практичному застосуванні це означає, що оператори продовжують витратити значну частину часу на ручну класифікацію запитів, незважаючи на високий рівень технологічної інтеграції платформи.

Іншим напрямом розвитку автоматизації є хмарні сервіси з елементами генеративного аналізу, серед яких Amazon Connect відіграє помітну роль. Завдяки інтеграції з Amazon Lex та модулями транскрипції Amazon Transcribe, ця система створює можливість побудови голосових та текстових ботів зі сценаріями, заснованими на NLP [11,12,13]. Однак досвід використання таких сервісів у багатомовному середовищі свідчить, що їхня точність залежить від обсягу якісних тренувальних даних та мовної підтримки. Для української мови ця підтримка є частковою, тому моделі можуть помилятися при обробці суржику, діалектів або неформальних формулювань. Внаслідок цього автоматизація в реальних бізнес-процесах продовжує залишатися асистивною, а не заміщуючою. Інтерфейс Amazon Lex зображено на рисунку 1.2.

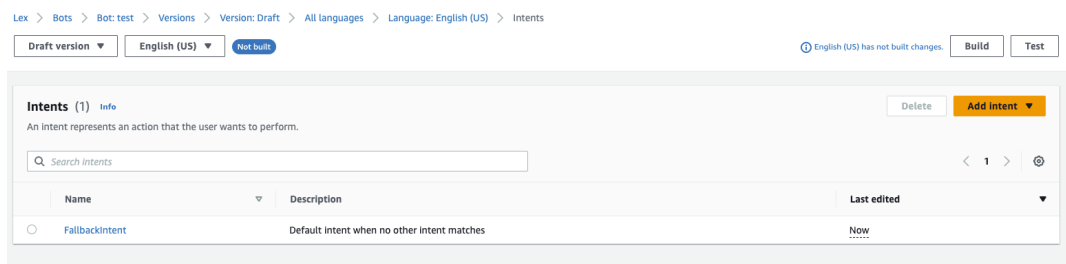


Рис. 1.2. Веб-інтерфейс Amazon Lex

У сфері чат-ботів важливе місце займають платформи Google Dialogflow CX, Microsoft Bot Framework, Rasa та інші фреймворки, орієнтовані на розробку діалогових систем. Dialogflow CX використовує прив'язку до ієрархії намірів, контекстів та станів, пропонуючи розширені можливості керування складними сценаріями [14,15]. Проте, подібно до інших великих екосистем, він демонструє найкращу ефективність на англійській мові. Для українських запитів точність класифікації варіюється і сильно залежить від специфічних шаблонів та додаткових правил, що зменшує універсальність моделі. Rasa, навпаки, дозволяє тренувати власні моделі, але вимагає значного обсягу якісного датасету, ретельної розмітки й глибокого розуміння внутрішньої архітектури.

Альтернативний клас рішень включає rule-based чат-боти та скриптові системи самообслуговування. У таких системах відповіді ґрунтуються на жорстко визначених патернах, ключових словах або виборі зі списку варіантів. Попри простоту реалізації, ці системи не здатні реагувати на складні або неоднозначні запити. Їхня ефективність суттєво знижується у випадках, коли клієнт використовує емоційні формулювання, жаргон, змішані мовні конструкції чи неелементарні питання. У певних сферах rule-based рішення можуть залишатися корисними, проте їхня обмеженість стає надто помітною в ситуаціях, де потрібне гнучке розуміння природної мови.

Окремо варто наголосити на технологіях IVR, які залишаються одним із найпоширеніших засобів автоматизації, проте їхня ефективність постійно знижується. Користувачі дедалі частіше уникають взаємодії з IVR через жорстку структуру меню, необхідність проходити послідовні рівні вибору та неможливість швидко передати суть проблеми. Незважаючи на локальні

покращення, ці системи не здатні до семантичного аналізу і фактично лише виконують маршрутизацію, не розв’язуючи проблему інтерпретації намірів.

Таблиця 1.2

Порівняння платформ автоматизації кол-центрів

Характеристика	Genesys Cloud CX	Amazon Connect	Google Dialogflow CX
Призначення	Оmnіканальний контакт-центр для великих організацій	Гнучкий хмарний контакт-центр з підтримкою кастомних AI-моделей	Платформа для побудови діалогових ботів та сценаріїв
NLP можливості	Обмежені, фокус на англійській	Гнучкі	Потужні, але зі зниженою точністю для української
Підтримка української мови	Низька	Середня (через кастомні моделі)	Часткова. залежить від обсягу тренувальних даних
Гнучкість налаштування	Висока у бізнес-процесах, низька в NLP	Дуже висока. повна кастомізація	Висока у діалогах, середня в інтеграції ML
Оптимальне застосування	Великі контакт-центри з класичною архітектурою	Організації, яким потрібні власні ML/NLP рішення	Проекти з акцентом на чат-ботах і діалогах

Підсумовуючи наведений аналіз, можна зробити висновок, що хоча ринок автоматизації контакт-центрів пропонує велику кількість інструментів, більшість із них зіштовхується з фундаментальними обмеженнями. Технології або недостатньо гнучкі, або вимагають великих ресурсів для налаштування, або не мають якісної підтримки української мови. Саме тому створення власної моделі визначення інтенцій, спеціально адаптованої під україномовний сегмент електронної комерції, є виправданим і актуальним напрямом дослідження. З огляду на реальні обмеження існуючих платформ, розробка локальної моделі класифікації намірів дозволить забезпечити точність, контроль якості та адаптацію до специфіки реальних клієнтських діалогів.

1.3. Постановка задачі дослідження

З огляду на виявлені у попередніх підрозділах проблеми сучасних кол-центрів та обмеження існуючих платформ автоматизації, виникає необхідність у створенні системи, здатної точно інтерпретувати природну мову користувача у сфері електронної комерції українською мовою. Центральним елементом такої системи є механізм автоматичного визначення інтенції (намірів) клієнта, оскільки саме класифікація запиту визначає подальший шлях його обробки, обрання сценарію відповіді та можливість часткового або повного усунення навантаження з операторів. У цьому контексті формулювання задачі дослідження потребує чіткого визначення ролі NLP-модуля, структуризації типових намірів та створення лінгвістичного підґрунтя для навчання моделі.

Відповідно до предметної області, що охоплює електронну комерцію, користувацькі звернення мають виражену варіативність, включають розмовні конструкції, емоційні вислови та елементи суржику. Це створює додаткову складність для систем автоматичної обробки тексту, оскільки традиційні rule-based або шаблонні рішення не здатні забезпечити коректну інтерпретацію таких запитів. Тому важливим етапом дослідження стало визначення набору найбільш поширених інтенцій, які відображають типові проблеми та потреби клієнтів онлайн-магазинів. Сформований нами перелік інтенцій — таких як запит статусу замовлення, перенесення часу доставки, скасування замовлення, питання щодо оплати, повернення товару, отримання інформації про продукт, проблеми з обліковим записом, питання доставки, застосування промокодів та звернення до оператора — дозволяє структурувати комунікативний простір користувацьких звернень та створює основу для формування навчального корпусу.

Постановка задачі передбачає не лише визначення цих інтенцій, але й побудову відповідного датасету, який репрезентує реальні мовні ситуації. Для дослідження було важливо сформувати достатньо великий корпус прикладів, що включає варіативні формулювання, різний стиль мовлення, різну довжину

висловлювань та можливі помилки або суржик. Це стало критичним чинником для навчання моделі класифікації, оскільки штучно створені або надто формальні приклади не відображають реальних особливостей комунікації між клієнтом і сервісом підтримки.

У межах даної роботи задача дослідження полягає у розробленні та експериментальному аналізі моделі машинного навчання, здатної визначати інтенцію вхідного текстового запиту українською мовою з використанням отриманого датасету. Модель має продемонструвати стабільність роботи в умовах мовної варіативності, нечіткості запитів, розмовних конструкцій та потенційних неточностей у формулюваннях. Особлива увага приділяється можливості узагальнення — система не повинна покладатися лише на чіткі шаблони, а має здатність інтерпретувати нові фрази, які не входили до тренувальної вибірки.

Таким чином, постановка задачі включає розроблення повного циклу NLP-компонента: від визначення інтенцій і формування мовного корпусу до побудови, навчання та оцінювання моделі класифікації. Це забезпечує основу для створення інтелектуального чат-боту, здатного автоматично інтерпретувати наміри користувачів та ефективно підтримувати роботу кол-центру у сфері електронної комерції.

1.4. Вибір засобів реалізації

Вибір технологічних засобів для реалізації системи автоматичного визначення інтенцій користувачів визначався сукупністю вимог, що впливають із поставленої задачі дослідження. Оскільки метою роботи є створення прототипу модуля інтелектуальної обробки текстових звернень українською мовою, ключову роль відіграли інструменти, здатні забезпечити гнучкість під час експериментування, підтримку сучасних методів обробки природної мови та можливість подальшої інтеграції у реальну програмну систему.

Основною платформою для побудови та навчання моделей було обрано мову програмування Python, яка є стандартом де-факто у галузі машинного

навчання завдяки своїй екосистемі бібліотек та фреймворків. Зокрема, у роботі використано бібліотеки PyTorch та інструментарій Hugging Face Transformers [16], що забезпечують зручний доступ до сучасних трансформерних архітектур та дозволяють виконувати тонке налаштування моделей під специфіку україномовного датасету. Застосування цих засобів надало можливість швидко створювати базові моделі, досліджувати їхню поведінку, проводити порівняльний аналіз та оптимізацію параметрів. Використання Python також забезпечило підтримку масштабного експериментування зі структурою датасету, попередньою обробкою тексту та формуванням вибірок.

Для підготовки даних, організації корпусу навчальних фраз та проведення моделювання було використано формат JSONL, який дозволяє ефективно працювати з великими текстовими вибірками, забезпечує зручність потокової обробки та добре інтегрується з наявними ML-інструментами. Розроблений датасет включав як синтетично згенеровані приклади, так і варіативні структурні конструкції, що відображають реальні запити у сфері електронної комерції. Це дозволило моделі ефективно адаптуватися до мовної динаміки та неоднорідності вхідних даних.

Окрему роль у структурі програмної системи відіграє Node.js, який було обрано як платформу для реалізації мікросервісу, що відповідає за бізнес-логіку та взаємодію з NLP-модулем. Node.js забезпечує високу продуктивність при обробці великої кількості одночасних текстових запитів та дозволяє інтегрувати Python-модель у вигляді окремого сервісу або API-ендпоінтів. Завдяки цьому NLP-компонент може бути використаний у складі ширшої системи чат-бота або контакт-центру, що відповідає вимогам масштабованості та розширюваності.

Для зберігання службових даних, журналів класифікацій або налаштувань системи було обрано MongoDB як документно-орієнтовану базу даних, яка забезпечує гнучку структуру та ефективний запис неструктурованої інформації. Використання MongoDB відповідає характеру даних, що

виникають у системах обробки діалогів, де формат збережених об'єктів може варіюватися.

Архітектурна модель системи розглядалася з урахуванням можливості розгортання в AWS, що забезпечує гнучкі сценарії масштабування, використання контейнеризації та інтеграцію з інструментами автоматизації. Платформа AWS дозволяє розміщувати як саму NLP-модель (наприклад, у вигляді сервісу на AWS Lambda або ECS), так і мікросервіс на Node.js, забезпечуючи їхню взаємодію через API-шлюзи. Використання інфраструктури AWS також відповідає сучасним тенденціям перенесення обчислювально складних модулів до хмарних середовищ.

Таким чином, обрані засоби реалізації формують узгоджену технологічну основу для розробки системи класифікації інтенцій користувачів. Python забезпечив можливості для створення та навчання моделей, Node.js — для організації бізнес-логіки та API-комунікації, MongoDB — для зберігання структурованих і напівструктурованих даних, а AWS — для потенційного масштабування та розгортання системи. У сукупності ці інструменти створюють придатну платформу для проведення повноцінного експериментального дослідження та подальшої інтеграції результатів у практичні рішення у сфері автоматизації кол-центрів.

Висновки до розділу 1

У першому розділі було узагальнено сучасний стан автоматизації кол-центрів і виявлено, що ключовою проблемою залишається точне розпізнавання намірів користувачів у природній мові, особливо в україномовному сегменті електронної комерції. Традиційні засоби автоматизації, такі як rule-based системи, IVR чи скриптові чат-боти, не забезпечують необхідної гнучкості та не здатні коректно інтерпретувати варіативні та неформальні мовні конструкції.

Порівняльний аналіз сучасних платформ — Genesys Cloud CX, Amazon Connect і Google Dialogflow CX — показав, що попри широкі можливості, їхнє

застосування для української мови є обмеженим, що унеможлиблює досягнення високої точності класифікації запитів без додаткової адаптації або створення власних рішень. Це підтверджує доцільність розроблення спеціалізованої NLP-моделі, орієнтованої на домен електронної комерції.

У межах постановки задачі було визначено набір інтенцій користувачів, характерних для типових звернень до онлайн-магазинів, а також сформовано мовний корпус для подальшого навчання моделі. Обґрунтовано вибір засобів реалізації, серед яких Python, PyTorch, бібліотека Hugging Face Transformers, Node.js та MongoDB, що забезпечують необхідну інфраструктуру для створення та інтеграції прототипу системи.

Таким чином, результати розділу сформували необхідне підґрунтя для розроблення та експериментальної перевірки моделі класифікації інтенцій, що розглядатиметься у наступних частинах роботи.

РОЗДІЛ 2. МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ НАМІРІВ КОРИСТУВАЧІВ

2.1. Методи побудови моделі визначення намірів

Задача визначення намірів користувача розглядається як процес віднесення текстового повідомлення до однієї з наперед визначених категорій, що відображають типові сценарії взаємодії у сфері електронної комерції. Формально вона належить до класу багатокласової класифікації, у межах якої для кожного вхідного висловлювання x необхідно визначити інтенцію $\hat{y}$, що максимізує умовну ймовірність

$$\hat{y} = \arg \max_{y \in Y} P(y | x)$$

Метою моделі є побудова наближення цієї ймовірнісної залежності на основі прикладів, що репрезентують характерні мовні патерни користувачів. У межах даного дослідження було застосовано два підходи до розв'язання задачі класифікації намірів. Перший з них — базовий статистичний метод, що ґрунтується на перетворенні тексту у векторну форму за допомогою TF-IDF та подальшому навчанні лінійного класифікатора. Такий підхід дозволяє зіставляти повідомлення на основі їхнього словникового складу та вагомості окремих термінів. Він демонструє прийнятну якість у випадках, коли формулювання запитів є відносно структурованим, однак практично не враховує семантичний контекст, порядок слів та прагматичні аспекти мови. Для україномовних користувацьких текстів, характерних для роботи кол-центрів, це створює певні обмеження через значну варіативність лексичних конструкцій, наявність розмовних форм, скорочень, опечаток та змішаних мовних шаблонів. Базова модель у нашій роботі використовується як порівняльна та дозволяє оцінити різницю між традиційними статистичними методами та сучасними нейромережевими моделями.

Другий, основний підхід базується на застосуванні трансформерної архітектури, що на сьогодні є найефективнішим інструментом для аналізу природної мови. У реалізації цієї роботи використано багатомовну модель

XLM-RoBERTa-base [17], попередньо натреновану на великому корпусі текстів понад ста мов, включно з українською. Це забезпечує значно глибше розуміння контексту, ніж у традиційних моделей, оскільки трансформер розглядає текст як взаємозалежну послідовність, у якій кожне слово може впливати на інтерпретацію будь-якого іншого слова. Центральним елементом такої архітектури є механізм самоуваги, який визначає ступінь важливості кожного токена для формування загального змісту фрази. Цей механізм описується виразом

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

де матриці Q, K та V формуються з прихованих представлень токенів. Завдяки цьому модель може однаково якісно обробляти як короткі, так і довгі повідомлення, коректно інтерпретувати неповні або емоційні фрази, визначати роль ключових слів у складних конструкціях та зберігати стійкість до мовних відхилень, зокрема у випадках суржику.

У процесі роботи модель XLM-RoBERTa була адаптована шляхом тонкого донавчання (fine-tuning) на сформованому україномовному датасеті, що містить фрази, характерні для реальних звернень до кол-центрів. Для реалізації було використано бібліотеку Hugging Face Transformers, яка надає високорівневі інструменти для роботи з попередньо навченими моделями, а також фреймворк PyTorch, що забезпечує можливість гнучкого налаштування процесу навчання, оптимізації та батчової обробки даних. Таким чином, модель отримала здатність розпізнавати специфічні мовні патерни, притаманні українським користувачам, і коректно класифікувати різні типи звернень у межах визначених інтенцій.

Застосування трансформерної архітектури у порівнянні зі статистичним baseline-підходом дозволяє досягти суттєвого підвищення точності, оскільки модель враховує не лише окремі слова, а й складні семантичні залежності між ними. Попереднє навчання на багатомовному корпусі також означає, що навіть за відносно невеликого обсягу доменного датасету модель демонструє високу

узагальнюючу здатність та стійкість до варіацій формулювань. Ці властивості роблять XLM-RoBERTa найбільш придатним інструментом для вирішення задачі класифікації намірів у сучасних системах підтримки клієнтів.

Сукупність двох підходів, базового статистичного та трансформерного, дозволяє всебічно оцінити ефективність сучасних методів NLP та підтвердити переваги моделей з механізмом самоуваги для задач, пов'язаних із реальними україномовними даними. Такий комбінований підхід забезпечує методологічну повноту дослідження та створює надійну основу для подальших етапів розроблення системи автоматизації обробки запитів користувачів.

2.1.1. Створення датасету

Ефективність моделі визначення намірів значною мірою залежить від якості та різноманітності тренувального датасету, який повинен репрезентувати реальні умови функціонування кол-центру у сфері електронної комерції. Оскільки відкриті україномовні корпуси звернень користувачів практично відсутні, виникла необхідність створити повністю власний датасет, у якому було б відображено природні формулювання клієнтів, типові помилки, емоційні конструкції, різні стилі мовлення та специфічні патерни поведінки. В межах роботи було розроблено процедуру генерації текстів із використанням API великих мовних моделей, що дозволило отримати широке покриття можливих варіацій запитів для кожної інтенції.

Початковим етапом стала формалізація переліку інтенцій, що відповідають домену електронної комерції. Були визначені такі категорії, як запити щодо статусу замовлення, зміни часу доставки, скасування замовлення, проблеми оплати, інформація про товар, робота промокодів, питання авторизації, повернення товарів, варіанти доставки та прохання з'єднати з оператором. Для кожної інтенції було сформовано набір базових seed-фраз, які використовувалися як джерело семантики і стилістичних орієнтирів для

подальшої генерації. Ці фрази були ретельно підібрані так, щоб охопити як нейтральні, так і емоційні сценарії взаємодії користувачів.

Подальший процес генерації базувався на Node.js-скриптах, які автоматизували взаємодію з OpenAI API [18,19]. Загальний принцип полягав у формуванні запитів до мовної моделі таким чином, щоб вона створювала одне унікальне речення за один виклик API. Такий підхід, на відміну від масової генерації великих фрагментів тексту, дозволив забезпечити максимальну різноманітність і уникнути надмірних повторів або шаблонності конструкцій. У тілі промπτу одночасно задавалися seed-фрази, короткі рекомендації щодо стилю та вказівки щодо обмеження на використання надто подібних формулювань. Для контролю унікальності скрипт зберігав усі отримані фрази та не дозволяв моделі генерувати повтори, перевіряючи кожен новий рядок перед додаванням до датасету. Функція для генерації запитів до OpenAI наведена в додатку А.

Особливу увагу було приділено реалістичності мовного матеріалу. З цією метою у промптах додатково регулювалися такі параметри, як довжина фрази, рівень ввічливості, емоційний стан користувача, а також можливість використання суржику. Останній є важливим елементом моделювання україномовного середовища, оскільки значна частина реальних клієнтських звернень має змішані мовні форми або відхилення від літературної норми. Параметр температури генерації було підібрано таким чином, щоб модель частіше пропонувала нестандартні конструкції, але водночас зберігала семантичну відповідність інтенції. Додатково застосовувалися коефіцієнти присутності та частотності, які зменшували ймовірність повторення однакових шаблонів.

Оскільки структура реальних звернень є нерівномірною, датасет також включає варіації з частковими або неповними формулюваннями, короткими емоційними репліками, уточнюючими питаннями та повідомленнями без контекстних елементів. Наприклад, запит «А де моє замовлення?» не містить номера замовлення, тоді як інші повідомлення включають уточнення, час

очікування або емоційну реакцію. Таке різноманіття покращує стійкість моделі до мовних відхилень, а також відображає реальні умови роботи підтримки клієнтів.

Для зберігання готових фраз було обрано формат JSONL, що є зручним для подальшої обробки у Python та бібліотеці Hugging Face. Кожний рядок містив об'єкт із двома ключами: текстом повідомлення та відповідним інтенційним класом. Кожна інтенція зберігалася в окремому файлі, що забезпечило зручність аналізу, перевірки якості та подальшого об'єднання у загальний корпус. Перед формуванням фінального датасету файли були об'єднані та перемішані, щоб уникнути упередженості моделі через можливу послідовність або групування прикладів за інтенціями.

Окремою задачею була перевірка якості згенерованих фраз. Частина з них відбраковувалася вручну у випадках, коли модель надто захоплювалася повтореннями або створювала фрази, які не відповідали інтенції через надмірну креативність. Така валідація дозволила забезпечити високу точність кінцевого датасету. У процесі роботи було створено приблизно рівномірну кількість фраз для кожної інтенції, що дозволило уникнути дисбалансу класів під час навчання моделі XLM-RoBERTa.

Таким чином, процес створення датасету включав кілька ключових етапів: визначення інтенцій, проектування seed-фраз, розробку генераційного скрипта, автоматизоване отримання унікальних зразків через API, контроль якості даних та їхнє структуроване збереження. Отриманий корпус охоплює широкий діапазон реалістичних клієнтських формулювань і є придатним як для навчання класичних методів, так і для сучасних трансформерних моделей. Він відображає різноманітність мовних патернів, властивих українським користувачам, і створює необхідну основу для побудови надійної системи класифікації намірів у рамках даного дослідження.

2.1.2. Конфігурація та навчання моделі

Створений у межах дослідження датасет став основою для формування та навчання моделей класифікації намірів, а відтак правильна конфігурація як

даних, так і самої моделі відіграє ключову роль у забезпеченні високої точності. Згенерований корпус фраз, отриманий за допомогою мовної моделі OpenAI, містив різноманітні приклади реальних користувацьких звернень, що включали нейтральні, емоційно забарвлені та інколи неповні формулювання. Завдяки спеціально створеним промптам кожне речення формувалося як унікальне, з урахуванням природних коливань довжини тексту, рівня ввічливості та наявності змішаних мовних конструкцій. Після генерації всі дані було збережено у форматі JSONL [20], що дало змогу легко інтегрувати їх у подальший процес навчання.

У ході підготовки дані було об'єднано в єдиний корпус, перемішано та перевірено на наявність фраз, які могли містити дублікати, неправильно визначену інтенцію або неприродні шаблонні конструкції. Особливу увагу було приділено збалансованості класів, оскільки нерівномірний розподіл прикладів впливає на точність класифікації. Для baseline-моделі дані було додатково перетворено за допомогою TF-IDF-векторизації [21], яка забезпечила числове представлення тексту, що зберігає інформацію про частотність термінів та їх релевантність у межах корпусу. Підготовка даних виконується програмним кодом наведеним у Додатку Б.

Першим етапом навчання став запуск сценарію `train_intent_baseline.py`, який реалізує традиційну методику обробки тексту. Логістична регресія була обрана як класифікатор з огляду на її стабільність та інтерпретованість. У процесі конфігурації baseline-моделі вибирали параметри векторизатора, такі як максимальний розмір словника, обмеження на мінімальну кількість появ слова та режим нормалізації. Це дозволило створити оптимізоване числове представлення текстів, яке зменшувало шум та покращувало здатність моделі виокремлювати релевантні ознаки. У логістичній регресії додатково задавалися параметри регуляризації, що запобігають перенавчанню, а також спосіб оптимізації, який забезпечував збіжність моделі навіть у випадку високої розмірності ознак. Тренування традиційної моделі наведено у Додатку В.

Другий етап передбачав навчання трансформерної моделі у сценарії `train_intent_transformer.py`, де використано архітектуру XLM-RoBERTa-base, яка характеризується високою чутливістю до контексту та багатомовною природою. Конфігурація трансформера складалася з кількох ключових аспектів. Насамперед було визначено токенізатор, який перетворював фрази на послідовності підслів, що є оптимальною одиницею для моделей сімейства RoBERTa. Далі задавалися параметри навчального циклу, включно з розміром батчу, кількістю епох та початковою швидкістю навчання. Для оптимізації ваг було використано алгоритм AdamW [22], який добре підходить для трансформерних моделей завдяки можливості контролю над регуляризацією та стабільністю градієнтів. Тренування трансформерної моделі наведено у Додатку В.

Особливу роль відігравала конфігурація `scheduler-a`, який поступово зменшував швидкість навчання у ході тренувального циклу. Це дозволило уникнути великих коливань на пізніх етапах навчання та підвищило точність розпізнавання інтенцій. Було визначено поділ датасету на тренувальну та валідаційну вибірки для постійного моніторингу якості та запобігання перенавчанню. Модель автоматично зберігала найкращі ваги згідно з результатами на валідаційній частині даних, що забезпечило стабільність фінальної версії класифікатора.

Під час навчання трансформера також застосовувалися конфігурації щодо максимальної довжини послідовності, відкидання значень паддінгу та коректної роботи attention-масок. Це дозволяло уникнути надмірного використання пам'яті та забезпечувало оптимальний розподіл уваги між релевантними частинами тексту. Налаштування `batch size` стали компромісом між обчислювальними можливостями та потребою у якісному градієнтному оновленні, тоді як `learning rate` визначав швидкість адаптації ваг до доменного датасету.

У сукупності конфігурація обох моделей — baseline та трансформерної — дозволила забезпечити повноцінний експериментальний цикл, у якому відображено еволюцію підходів до обробки природної мови: від класичних статистичних методів до сучасних контекстуальних архітектур. Трансформерна модель, будучи адаптованою до українського мовного середовища за допомогою тонкого донавчання, продемонструвала здатність гнучко реагувати на різні стилі користувацьких звернень, тоді як baseline-модель виконувала роль контрольної, дозволяючи оцінити приріст ефективності. Таким чином, конфігурація та навчання моделей сформували методологічну основу дослідження та забезпечили надійну платформу для подальшого аналізу точності, узагальнювальної здатності та стійкості системи класифікації намірів до реальних мовних даних.

2.2. Аналіз отриманих даних

Отримані результати навчання двох моделей — базової (Logistic Regression + TF-IDF) та трансформерної XLM-RoBERTa — дали змогу оцінити ефективність різних підходів до задачі класифікації намірів користувача в системі підтримки контакт-центру. Аналіз даних охоплює порівняння точності для кожного інтенду, оцінку F1-показників, а також дослідження типових помилок за допомогою матриць змішувань.

Базова модель продемонструвала загальну точність на тестовій вибірці 0.96, що вже свідчить про адекватність обраного підходу TF-IDF + Logistic Regression для класифікації коротких фраз українською та суржиком. Незважаючи на простоту методу, модель коректно класифікує більшість інтендів, що підтверджується відсутністю систематичних помилок у більшості класів. Лише окремі інтенди, зокрема `payment_issue` та `promo_or_discount`, інколи демонструють локальні змішування, імовірно спричинені семантично близькими формулюваннями користувачів.

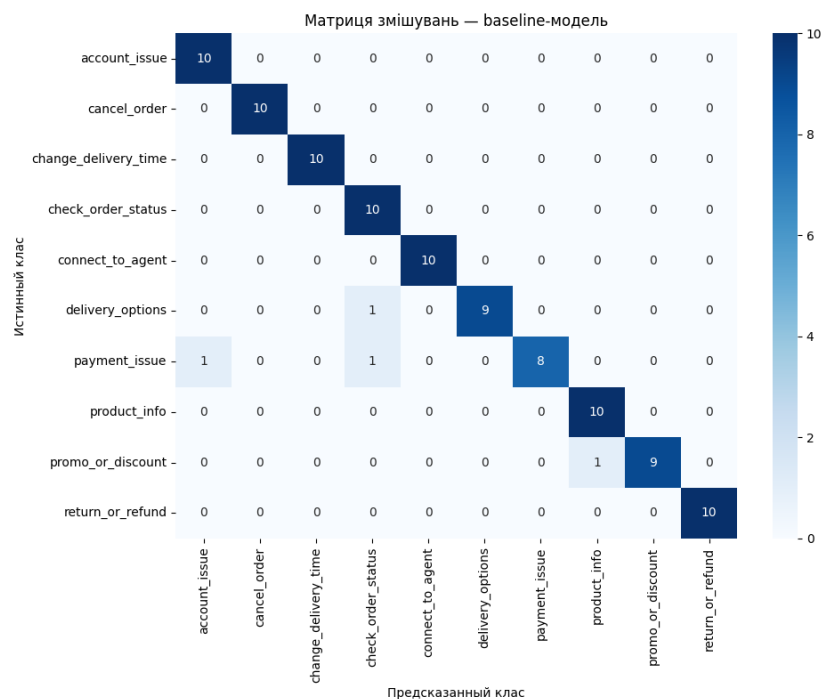


Рис. 2.1. Матриця змішувань для baseline-моделі

Поданий графік матриці змішувань демонструє, що хибні класифікації трапляються спорадично та не становлять значної загрози для роботи системи. Наприклад, окремі фрази з фінансовим контекстом можуть бути інтерпретовані як запити на отримання промокоду або навпаки, що відображає природну лінгвістичну неоднозначність.

Порівняльний аналіз F1-показників свідчить, що для більшості інтендів значення перевищує 0.9, а для кількох досягає максимальної оцінки. Це підтверджує хорошу узагальнювальну здатність baseline-моделі попри її простоту.

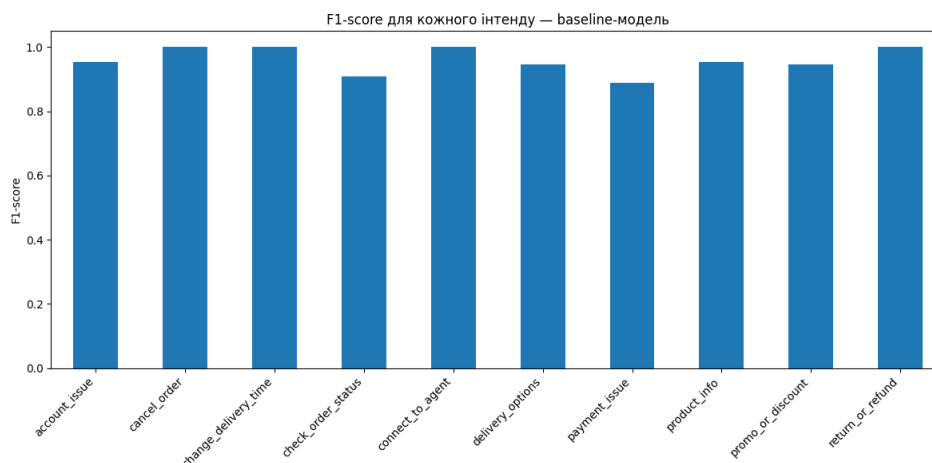


Рис. 2.2. F1-score для кожного з намірів для baseline-моделі

Перехід до трансформерної архітектури — XLM-RoBERTa — дозволив значно підвищити якість класифікації та зменшити кількість помилок. Модель показала майже безпомилкову класифікацію за більшістю інтендів, демонструючи точність та F1-показники на рівні, близькому до 1.0. Це очікуваний результат, оскільки XLM-R навчається на багатомовних корпусах, що робить її стійкою до мовних варіацій і орфографічних відхилень, характерних для реальних повідомлень користувачів.

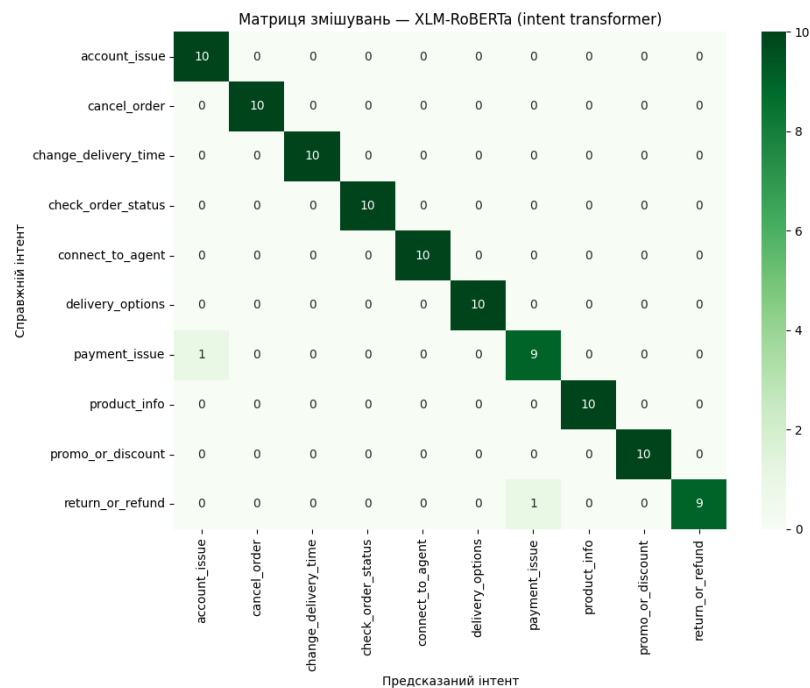


Рис. 2.3. Матриця змішувань для XLM-RoBERTa

Матриця змішувань для XLM-RoBERTa демонструє відсутність суттєвих систематичних зсувів. Невеликі розбіжності, які все ж трапляються, зумовлені, ймовірно, семантично близькими інтендами (наприклад, “payment_issue” та “return_or_refund”), однак їх масштаб значно менший порівняно з baseline-підходом.

Оцінювання F1-показників підтверджує перевагу трансформерної моделі: майже всі класи отримали значення $F1 \approx 1.0$, що свідчить про високу чутливість та специфічність класифікатора. Такий рівень якості дає змогу використовувати модель не лише для лабораторного експерименту, а й як

основу для реального промислового компонента інтелектуальної маршрутизації звернень у кол-центрі.

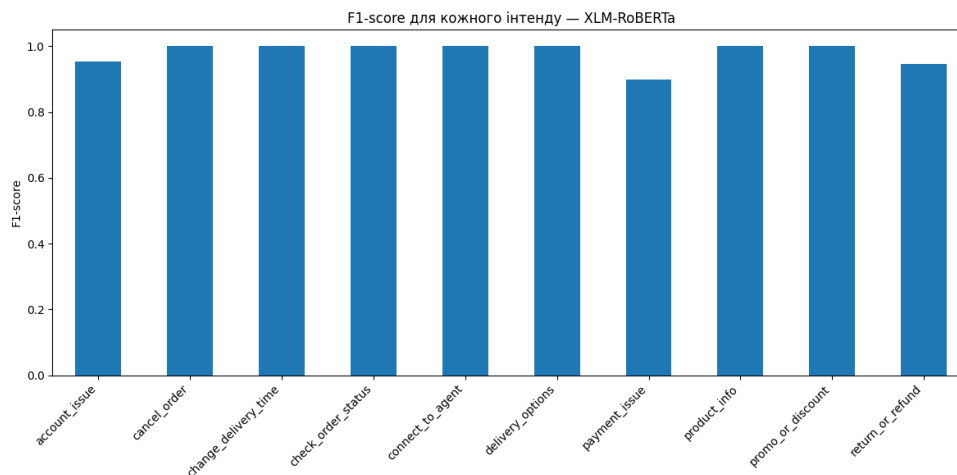


Рис. 2.4. F1-score для кожного з намірів для XLM-RoBERTa

Порівняння baseline-підходу та XLM-R свідчить, що обидві моделі є працездатними, однак їх потенційні сфери застосування різняться. Простий логістичний класифікатор може використовуватися як легковага альтернатива для систем із жорсткими обмеженнями на ресурси, проте для розгортання у ситуаціях, де важлива максимальна точність у багатомовному середовищі, доцільно застосовувати саме XLM-RoBERTa. Аналіз отриманих даних демонструє закономірне зростання якості при переході від класичних методів опрацювання тексту до сучасних трансформерних моделей.

2.3. Проектування додатку

Проектування архітектури системи для автоматизованого визначення інтенцій користувацьких запитів передбачає створення надійної, масштабованої та модульної платформи, яка здатна працювати у режимі реального часу. Оскільки інтентифікація запитів здійснюється за допомогою моделі машинного навчання, що потребує значних обчислювальних ресурсів, важливо забезпечити чітке розділення відповідальностей між компонентами системи. Кінцевою метою архітектурного проектування є формування такої структури сервісів, яка дозволяє легко адаптувати систему під зміни навантаження, оновлення моделі, розширення функціональності та інтеграцію нових механізмів обробки діалогів.

У побудованій системі клієнтським рівнем виступає Vue 3 [23] веб-застосунок, який забезпечує відображення інтерфейсу та взаємодію з користувачем. Він ініціює надсилання текстових повідомлень, підтримує встановлення WebSocket-каналу [24] та отримує відповіді у режимі реального часу. Двостороння комунікація особливо важлива для сценаріїв, де користувачу потрібно негайно повідомити про результати обробки або зміну статусу запиту. Саме тому рішення включає одночасне використання REST API та WebSocket-з'єднання, що підвищує швидкість та інтерактивність.

Серверна частина проєкту складається з двох основних сервісів, між якими чітко розподілено функції: Node.js gateway-сервіс та Python-сервіс визначення інтенцій. Gateway-сервіс реалізує REST-інтерфейс, приймає вхідні повідомлення, створює первинний запис у базі даних, надсилає текст запиту до сервісу визначення інтенцій та повертає відповідь користувачу через WebSocket. Такий підхід дозволяє ізолювати усю логіку маршрутизації та бізнес-процесів на одному шарі, що значно спрощує масштабування та супровід системи.

Python-сервіс працює як окремий мікросервіс, що завантажує попередньо натреновану модель XLM-RoBERTa та обробляє запити на визначення інтенцій. Його винесення в окремий контейнер дозволяє незалежно керувати обчислювальними ресурсами, оновленнями моделі та оптимізацією процесу інференсу. Це рішення зменшує затримку часу обробки та спрощує майбутнє розширення системи, наприклад, можливе застосування декількох моделей або впровадження ансамблевих підходів.

Зберігання даних реалізовано за допомогою MongoDB або Amazon DocumentDB, що дозволяє фіксувати історію запитів, журнал подій, результати класифікації та метадані взаємодії. Наявність централізованого сховища даних є критично важливою для моніторингу якості роботи системи, оптимізації моделі та подальшого удосконалення алгоритмів взаємодії з користувачем. Завдяки цьому можливо аналізувати розподіл інтенцій,

визначати складні для моделі запити, а також реалізувати аналітику на рівні бізнес-процесів.

Усі серверні компоненти проєкту планується розгортати у хмарній інфраструктурі AWS. Такий вибір зумовлений потребою забезпечити високу доступність, відмовостійкість та автоматичне масштабування під навантаженнями. Розміщення клієнтських статичних файлів у Amazon S3 разом із використанням CloudFront дозволяє значно прискорити доставку контенту користувачам. Gateway-сервіс і сервіс класифікації інтентів працюватимуть у контейнерах Amazon ECS Fargate, що забезпечує автоматичне масштабування та відсутність необхідності керувати серверною інфраструктурою безпосередньо. Application Load Balancer розподіляє запити між екземплярами контейнерів, запобігаючи перевантаженню окремих компонентів.

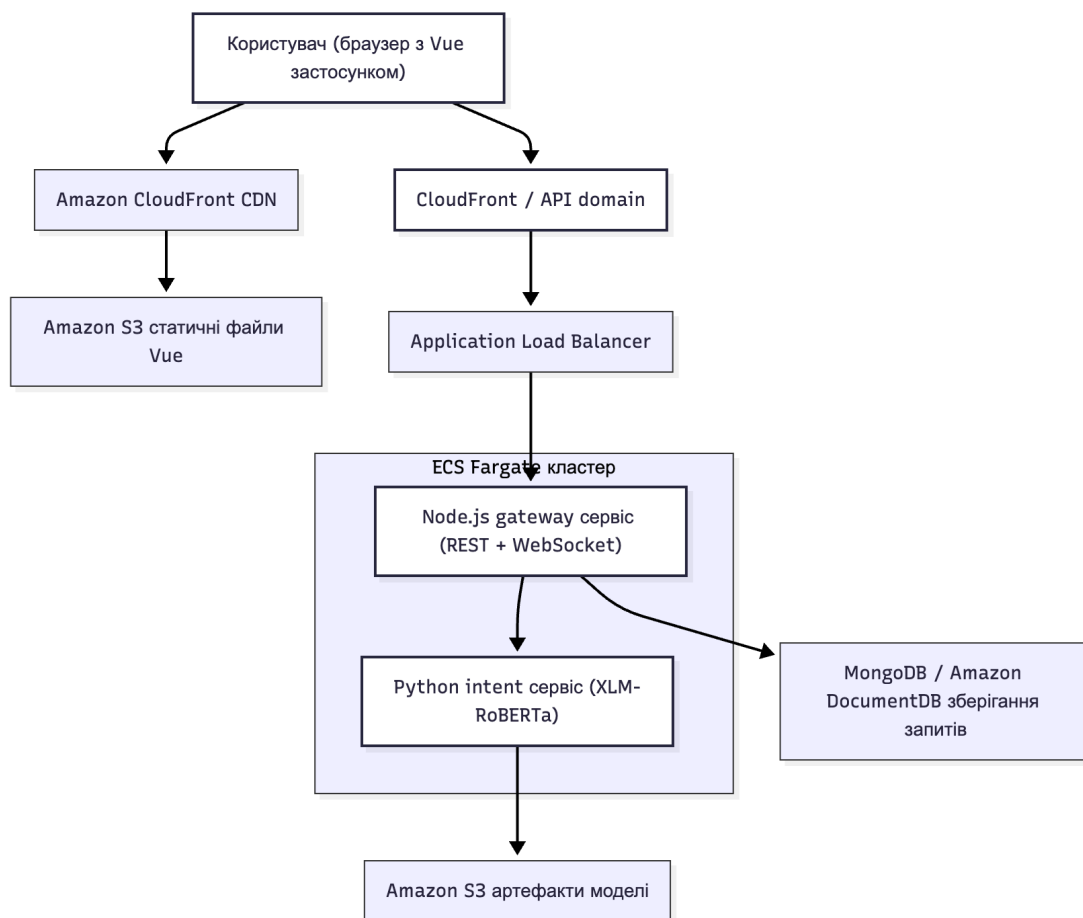


Рис. 2.5. Загальна архітектура розгортання системи у AWS

Побудована інфраструктура забезпечує окреме масштабування для кожного рівня системи. Наприклад, за високого навантаження на NLP-модель можна збільшити кількість контейнерів Python-сервісу без зміни конфігурації gateway-сервера. Так само фронтенд може обслуговувати практично необмежену кількість користувачів завдяки кешуванню через CloudFront. Така модульність сприяє покращенню продуктивності та стійкості системи у промисловому середовищі.

Потік взаємодії між компонентами системи починається з ініціалізації WebSocket-з'єднання клієнтом. Після цього користувач надсилає текстове повідомлення, яке gateway-сервіс реєструє у базі даних зі статусом received. Далі через REST-виклик текст передається у Python-сервіс, який визначає інтенцію та повертає ймовірнісний розподіл. Gateway-сервіс оновлює статус у базі, зберігає результат і надсилає інтерпретовану відповідь клієнту через WebSocket. Таким чином система підтримує асинхронний режим роботи, що є надзвичайно важливим для побудови інтелектуальних діалогових інтерфейсів.

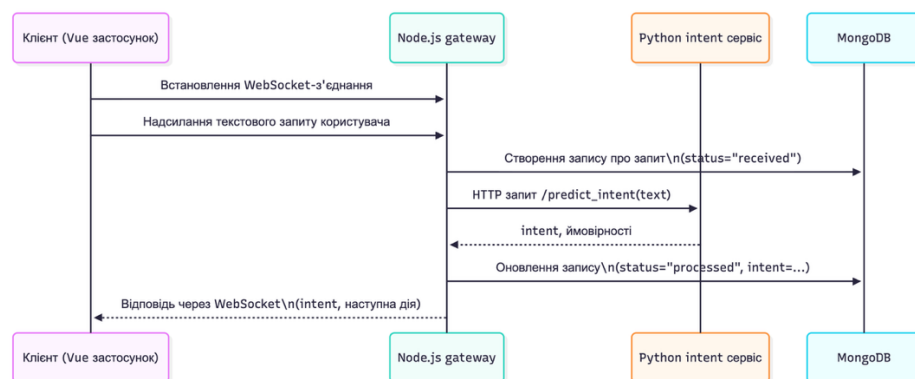


Рис. 2.6. Послідовність взаємодій між клієнтом та Python-сервісом

Логічний поділ компонентів системи демонструє збалансований підхід до організації мікросервісної архітектури. Фронтенд зосереджується виключно на взаємодії з користувачем, gateway — на логіці маршрутизації та координації сервісів, Python-модуль — на інференсі моделі, а база даних — на забезпеченні довготривалого зберігання інформації. Такий поділ дозволяє незалежно

розвивати кожний компонент, впроваджувати нові можливості та інтегрувати зовнішні сервіси, не порушуючи загальну структуру системи.

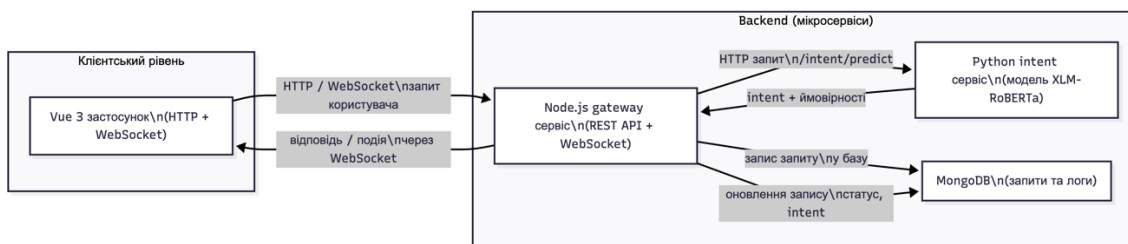


Рис. 2.7. Логічна архітектура клієнтського та серверного рівнів системи

Архітектура також закладає основу для майбутніх вдосконалень. Наприклад, у gateway-сервіс може бути інтегрований модуль діалогового менеджменту, який визначатиме наступні дії залежно від інтенції. Python-сервіс може бути розширений функціональністю для слот-філінгу або контекстної обробки. У перспективі можлива реалізація мультиагентної структури, де різні моделі вирішують частини задачі, або перехід до serverless-рішення на базі AWS Lambda для окремих операцій. Таким чином, створена система має високу гнучкість, яка дозволяє розвивати її як у напрямку складніших AI-алгоритмів, так і у напрямку інтеграції з бізнес-процесами підприємства.

Висновки до розділу 2

У другому розділі було здійснено комплексне опрацювання методів та технологій, необхідних для побудови системи автоматизованого визначення інтенцій користувацьких запитів. Проведений аналіз показав, що задача класифікації інтенцій належить до класу проблем обробки природної мови, які вимагають як високої якості представлення текстових даних, так і відповідної архітектури моделювання. У межах дослідження було розглянуто та обґрунтовано два підходи: базовий лінійний класифікатор на основі TF-IDF та логістичної регресії, а також трансформерну модель XLM-RoBERTa, що

забезпечує глибоке контекстуальне розуміння тексту. Порівняння цих підходів демонструє суттєву різницю у здатності до узагальнення, що підтверджує доцільність застосування сучасних трансформерних методів для вирішення задачі інтентифікації.

У розділі було показано, що формування датасету відіграє ключову роль у точності моделі. Було описано процес генерації прикладів за допомогою інструментів OpenAI, підготовки даних, маркування та структурування корпусу для навчання моделей. Особливу увагу приділено питанням збалансованості набору даних, різноманітності формулювань інтенцій та усуненню дублювань, що забезпечило якісне підґрунтя для моделювання.

Підсистема навчання моделі була представлена у вигляді двох окремих процесів — навчання baseline-класу та повноцінної трансформерної моделі. Було описано конфігурацію кожного методу, вибір гіперпараметрів, а також мотиви, що зумовили необхідність порівняння різних архітектур. Демонстрація покрокових процедур із навчання моделей дозволила оцінити їх працездатність у реальних умовах та визначити сильні сторони кожного підходу.

Завершальним етапом розділу було проектування архітектури системи, орієнтованої на промислове використання. Було розроблено логічну та інфраструктурну схеми, які демонструють взаємодію клієнта, gateway-сервісу, сервісу визначення інтенцій та бази даних. Запропоноване рішення побудоване з урахуванням вимог масштабованості, стійкості та поділу відповідальностей між мікросервісами. Використання Amazon Web Services дає змогу забезпечити високу доступність системи, а застосування WebSocket-каналу дозволяє реалізувати інтерактивний механізм обміну повідомленнями з мінімальною затримкою.

Отже, результати розділу формують комплексне підґрунтя для подальших експериментів та оцінювання ефективності моделей, що буде розглянуто у наступному розділі. Проведена методологічна та архітектурна підготовка створює цілісну основу для реалізації інтелектуальної системи

визначення інтенцій у кол-центрах та забезпечує умови для її подальшого вдосконалення й масштабування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МІКРОСЕРВІСНОЇ ІНФРАСТРУКТУРИ ЧАТБОТУ З МОДУЛЕМ ВИЗНАЧЕННЯ ІНТЕНЦІЙ

3.1. Основні вимоги до системи

Проектування системи автоматичного визначення намірів користувачьких запитів вимагає формулювання чітких функціональних та нефункціональних вимог, які визначають очікувану поведінку, межі застосування та критерії якості програмного забезпечення. Оскільки чатбот у контексті кол-центру виконує роль проміжної ланки між користувачем і сервісною інфраструктурою підприємства, він повинен забезпечувати не лише високу точність обробки природної мови, а й стабільність, передбачуваність реакцій та інтегрованість у вже існуючі бізнес-процеси. У цьому сенсі вимоги до системи охоплюють як алгоритмічні та модельні аспекти, так і вимоги до архітектури, продуктивності та експлуатаційних характеристик.

Основним функціональним завданням системи є здатність коректно класифікувати текстові звернення користувачів за заздалегідь визначеним набором інтенцій. Модель має забезпечувати стійкість до варіативності мовлення, включно з розмовними формами, помилками, скороченнями, суржилом та широким спектром контекстів реального використання. Оскільки чатбот використовується у сфері електронної комерції, класифікація повинна охоплювати типові запити клієнтів, пов'язані з доставкою, оплатою, поверненням товарів та потребою у зв'язку з оператором. Кожен такий запит має бути оброблений однозначно, із поверненням інтенції, яку система може використати для подальших сценаріїв діалогу. Крім того, система повинна надавати модельний прогноз у реальному часі, щоб користувач не відчував додаткових затримок у процесі взаємодії.

Окрему важливість становить вимога інтегрованості та модульності. Модель визначення інтенцій функціонує як окремий мікросервіс, тому інші компоненти системи — клієнтський застосунок, gateway-сервіс, база даних та зовнішні бізнес-процеси — повинні мати можливість взаємодіяти з нею через стандартизовані API. Такий підхід дозволяє незалежно оновлювати модель,

масштабувати її або замінювати більш досконалішими рішеннями без втручання в інші частини системи. Крім того, компоненти повинні обмінюватися даними у спосіб, що зберігає цілісність інформації та гарантує узгодженість результатів обробки запитів.

Важливим нефункціональним аспектом є продуктивність системи та її здатність працювати за умов підвищеного навантаження. Оскільки чатбот може одночасно обслуговувати значну кількість користувачів, система повинна підтримувати горизонтальне масштабування сервісу визначення інтенцій та gateway-рівня. Затримка відповіді моделі має залишатися низькою, навіть якщо кількість одночасних запитів зростає, що передбачає використання оптимізованих механізмів інференсу та розподілених обчислень. У межах AWS-інфраструктури ця вимога реалізується за допомогою контейнеризації (ECS Fargate), автоматичного масштабування та балансування навантаження через Application Load Balancer.

Окрему увагу необхідно приділити надійності та відмовостійкості системи. Оскільки чатбот є частиною сервісу, що взаємодіє з реальними клієнтами, недоступність NLP-сервісу або gateway-рівня може негативно вплинути на роботу підприємства. Саме тому всі критичні компоненти повинні мати можливість автоматичного перезапуску, а дані — зберігатися у стійких сховищах, здатних витримувати відмови окремих вузлів. У разі збою модель не повинна блокувати загальну роботу системи, а клієнт має отримати коректне резервне повідомлення або спрямування до оператора.

Безпека даних є невід’ємною вимогою сучасних інтелектуальних систем. Усі запити користувачів повинні передаватися через захищені канали, а зберігання даних не може допускати витоку персональної інформації. З огляду на це використання HTTPS, обмеження доступу до Python-сервісу, сегментація мереж та керування секретами у хмарному середовищі є обов’язковими елементами архітектури. Оскільки дані можуть містити персональні елементи (ім’я клієнта, номер замовлення, контактну інформацію), система повинна гарантувати їхню конфіденційність та недоступність для сторонніх осіб.

Ураховуючи специфіку проєкту, значну роль відіграє також вимога пояснюваності та відтворюваності результатів моделі. Оскільки модель може приймати рішення на основі складних контекстуальних закономірностей, важливо забезпечити можливість аналізу її роботи та оцінювання її продуктивності після оновлення або перевчання. Рішення має підтримувати логування запитів, фіксацію ймовірнісних розподілів, а також можливість подальшого аналізу цих даних у модулі аналітики. Завдяки цьому можна виявляти помилки класифікації, визначати потребу у розширенні навчального датасету та покращувати модель у наступних ітераціях.

Таким чином, основні вимоги до системи визначення інтенцій охоплюють широкий спектр аспектів — від точності та швидкодії моделі до надійності, безпеки та масштабованості всієї інфраструктури. Їх урахування на етапі проєктування дозволяє створити платформу, яка здатна працювати в умовах реального навантаження, забезпечувати стабільність обробки запитів та інтегруватися з існуючими сервісами кол-центру. Підготовлена система формує технічне підґрунтя для подальшого розширення функціональності чатботу та переходу до більш складних діалогових моделей.

3.2. Вибір інструментальних засобів розробки

Розроблення системи автоматичного визначення інтенцій у межах інтерактивного чатботу потребує ретельно обґрунтованого вибору інструментальних засобів, оскільки якість, масштабованість і подальша підтримка програмного комплексу суттєво залежать від технологічних рішень, використаних на етапі проєктування та реалізації. Оскільки система складається з кількох архітектурно незалежних частин — клієнтського застосунку, gateway-рівня, мікросервісу з нейронною моделлю та інфраструктурних складових, — вибір технологій формувався з огляду на їхню сумісність, стабільність, доступність інструментів розробки та здатність забезпечувати необхідну продуктивність у реальних умовах експлуатації.

Для клієнтської частини було обрано фреймворк Vue.js, який забезпечує високу швидкість рендерингу, реактивність інтерфейсу та оптимальний баланс між функціональністю та продуктивністю. Вебклієнт виконує ключову роль у взаємодії з користувачем, тому важливою була можливість створення динамічного інтерфейсу з підтримкою WebSocket-комунікації та асинхронного обміну даними. Vue.js відрізняється невеликою вагою, простотою адаптації під складні інтерфейси та розвиненою екосистемою, що дозволяє ефективно реалізувати всі необхідні функції чатботу.

На серверному рівні центральне місце займає gateway-сервіс, реалізований засобами Node.js. Вибір цієї платформи пояснюється її високою продуктивністю у роботі з мережевими запитами, підтримкою неблокуючої моделі введення-виведення та широким використанням у мікросервісних архітектурах. Gateway-сервіс виконує роль координатора, приймає вхідні повідомлення від клієнта, реєструє запити у базі даних, взаємодіє з Python-сервісом для визначення інтенції та негайно повертає відповідь користувачу через WebSocket. Node.js природно підходить для таких сценаріїв, оскільки спроможний обробляти велику кількість одночасних з'єднань з мінімальною затримкою та забезпечує гнучке масштабування.

Мікросервіс визначення інтенцій побудований на Python, що є де-факто стандартом у сфері машинного навчання та природної мови. Використання бібліотеки Transformers від Hugging Face забезпечило можливість швидкої інтеграції попередньо натренованої моделі XLM-RoBERTa та її донавчання на доменному датасеті. Python дозволяє здійснювати інференс моделі з мінімальними накладними витратами, має широкі можливості для обробки текстів та оптимізації продуктивності, а також значно полегшує підтримку та подальше вдосконалення моделей. Обраний інструментарій дає змогу ізолювати NLP-логіку в окремому сервісі, що забезпечує незалежність, гнучкість і можливість майбутньої заміни моделі без втручання в інші компоненти системи.

У межах наукової складової проєкту було також застосовано інструменти для моделювання та аналізу. Зокрема, PyTorch використовувався як основна бібліотека для роботи з нейронними мережами, оскільки забезпечує високий рівень контролю над процесом навчання та дозволяє ефективно виконувати обчислення як на CPU, так і на GPU. У випадку baseline-рішення головну роль відіграв scikit-learn, за допомогою якого реалізовано TF-IDF векторизацію та логістичну регресію. Порівняння цих двох підходів сформувало експериментальну основу дослідження.

Для зберігання даних обрано MongoDB [25], що відповідає вимогам до динамічної структури документів, наявності вкладених об'єктів та можливості швидкого збереження операцій журналювання. Нереляційна модель дозволяє без значних накладних витрат зберігати історію повідомлень, метадані та результати класифікації. У хмарному середовищі MongoDB або DocumentDB забезпечують автоматичне резервування, масштабованість та стійкість до відмов.

Організація інфраструктури базується на інструментах AWS, що дозволяють створювати гнучку та відмовостійку архітектуру. Розміщення контейнерів у Amazon ECS Fargate усуває потребу керування серверними вузлами та значно спрощує масштабування. Використання Application Load Balancer забезпечує ефективний розподіл навантаження між контейнерами з gateway-логікою та NLP-сервісом. Для статичних файлів інтерфейсу використовується Amazon S3 у поєднанні з мережею доставки контенту CloudFront, що гарантує мінімальну затримку та високу доступність. Застосування цих технологій дозволяє створити інфраструктуру, здатну працювати у промисловому середовищі з можливістю автоматичного реагування на зміни трафіку.

Загалом вибір інструментальних засобів зумовлений вимогами до гнучкості архітектури, швидкості обробки запитів, можливості поділу логіки на незалежні сервіси та забезпечення високої точності класифікації. Кожен компонент системи використовує технології, найбільш оптимальні для свого

функціонального призначення, що забезпечує збалансованість у продуктивності, стабільності та розширюваності. Завдяки такому підходу створене рішення поєднує сучасні підходи до розробки, машинного навчання та хмарної інфраструктури, формуючи основу для подальших експериментів і масштабування.

3.3. Реалізація окремих модулів чатботу

Реалізація чатботу з автоматичним визначенням намірів користувача ґрунтується на модульній структурі програмного забезпечення, що дозволяє розподілити відповідальність між окремими компонентами та забезпечити їх незалежний життєвий цикл. Кожен модуль системи виконує чітко визначену функцію, а взаємодія між ними здійснюється через стандартизовані інтерфейси. Такий підхід не лише спрощує підтримку та модернізацію системи, але й забезпечує можливість масштабування як обчислювальних ресурсів, так і логічних можливостей чатботу. Реалізована архітектура передбачає наявність сервісу визначення інтенцій, оркестраційного сервісу, підсистеми сценарної логіки, компонентів керування даними, а також двох окремих клієнтських застосунків — користувацького та адміністративного.

Фундамент системи становить модуль визначення інтенцій, реалізований у середовищі Python. Цей сервіс виконує обчислювально складне завдання передбачення наміру користувача за допомогою попередньо навченої моделі, побудованої на основі трансформерної архітектури. У файлі `inference.py` описано завантаження токенизатора та моделі, їх ініціалізацію у пам'яті процесу та логіку перетворення текстового запиту у числове подання, що подається на вхід моделі. Вся операція інференсу виконується локально в оперативній пам'яті, що мінімізує затримку під час оброблення численних послідовних запитів. Файл `service.py` реалізує обгортку у вигляді мікросервісу, що працює на базі FastAPI [26] і забезпечує HTTP-інтерфейс для зовнішніх компонентів системи. Таким чином, Python-модуль функціонує як ізольована обчислювальна одиниця, яка отримує текст та повертає відповідний інтент разом з імовірнісним розподілом.

Наступним ключовим компонентом є центральний Node.js-сервіс, який виконує роль оркестратора та об'єднує всі логічні елементи системи в єдиний робочий конвеєр. Файл `server.js` містить реалізацію механізмів оброблення WebSocket-з'єднань, створення нових сесій користувачів, маршрутизації подій та формування відповідей чатбота. Саме цей модуль отримує вхідне повідомлення, викликає Python-сервіс для визначення інтенції, обробляє відповідь моделі та передає її до наступного етапу опрацювання. Архітектура сервісу передбачає постійні WebSocket-канали між клієнтом та сервером, що забезпечує режим реального часу та швидку реакцію чатботу на дії користувача. Усі повідомлення зберігаються системою логування, що дозволяє підтримувати історію діалогів та забезпечує можливість їх подальшого аналізу.

Над визначенням інтенту розташована окрема підсистема сценарної логіки, реалізована у файлі `intentScenarios.js`. Її призначення полягає в інтерпретації результатів роботи моделі та формуванні відповідної поведінки чатботу. На основі визначеного інтенту ця підсистема приймає рішення щодо наступних кроків: повернення повідомлення, ініціації діалогу з уточнювальними питаннями, запуску умовної гілки сценарію або ескалації запиту до оператора. Кожен сценарій описується у структурованому форматі, що дозволяє додавати нові інтеракції без модифікації ядра серверної логіки. Цей підхід суттєво підвищує гнучкість системи та забезпечує можливість поступового розширення функціональності чатботу з мінімальними витратами.

Керування даними реалізовано через окремий модуль, описаний у файлі `database.js`, який забезпечує взаємодію з MongoDB. База даних використовується для збереження інформації про сесії, повідомлення, статуси оброблення та інші метадані, необхідні для коректної роботи системи. У межах цього компонента виконуються всі ключові операції з персистентністю даних: створення нових записів, оновлення параметрів поточних діалогів, завершення сесій та забезпечення доступу до історії звернень. Вибір

нереляційної бази даних виявився оптимальним з огляду на нерегулярну структуру повідомлень та необхідність швидкого збереження даних у режимі реального часу.

Реалізація клієнтської частини системи включає два окремі застосунки: інтерфейс користувача та адміністративну панель. Користувацький застосунок, побудований у межах проєкту chat-client, забезпечує основну взаємодію відвідувача з чатботом через WebSocket-канал. Він містить інтерфейс обміну повідомленнями, механізми відображення сценарних кроків та інформування користувача про можливість передачі діалогу оператору. Адміністративний застосунок chat-admin орієнтований на моніторинг та аналіз сесій, надає можливість переглядати історію повідомлень, статуси оброблення та загальну динаміку звернень. Обидва застосунки побудовані з використанням сучасних веб-технологій, що забезпечує високу інтерактивність та зручність роботи.

Таким чином, реалізація окремих модулів чатботу демонструє цілісну архітектуру, в якій кожен компонент виконує чітко визначені функції та взаємодіє з іншими через стандартизовані інтерфейси. Такий підхід забезпечує надійну роботу системи, ефективне оброблення повідомлень, гнучкість у розширенні сценарної логіки та можливість масштабування в хмарному середовищі. Реалізована система поєднує методи машинного навчання та інженерію програмних систем, утворюючи комплексне рішення, готове до подальшої інтеграції в екосистеми контакт-центрів.

3.4. Опис інтерфейсу програмного продукту

Розроблений клієнтський інтерфейс чат-боту виконує роль точки входу користувача в систему і забезпечує початкову взаємодію з сервісом визначення намірів. Основна логіка роботи побудована навколо поєднання HTTP-запитів та WebSocket-каналу, що дозволяє формувати інтерактивний досвід спілкування в режимі, близькому до реального часу. Усі ключові етапи взаємодії — встановлення з'єднання, передавання повідомлень, отримання

класифікації наміру та повернення відповіді — відображено у поведінці UI, що подано на відповідних ілюстраціях.

Після відкриття застосунку користувач спочатку бачить вітальне вікно із запитом ввести ім'я. Цей крок не є лише декоративним елементом, а забезпечує ініціалізацію індивідуальної сесії, що дозволяє надалі асоціювати всі дії користувача з окремим логічним контекстом. Саме на цьому етапі відбувається встановлення WebSocket-з'єднання з Node.js-gateway, який виступає центральним комунікаційним вузлом між клієнтом та Python-сервісом класифікації.

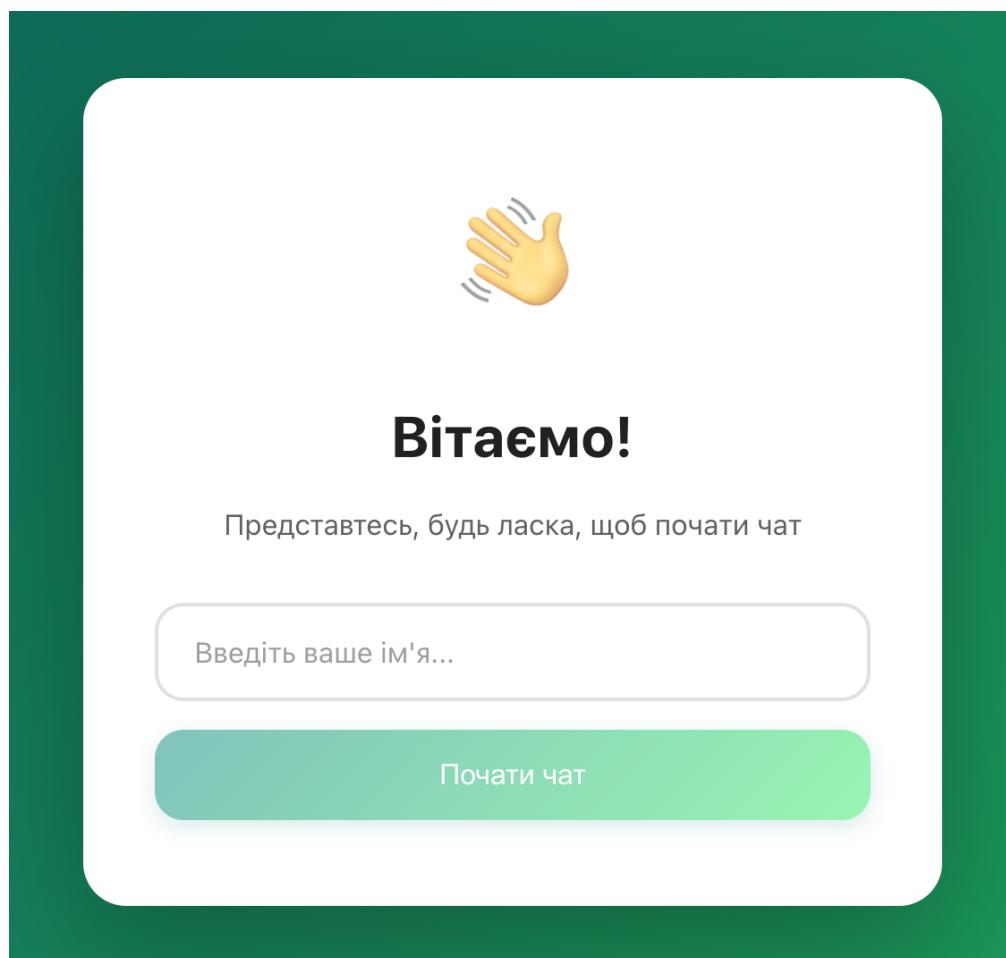


Рис. 3.1. Вікно введення ім'я користувача

Після введення імені чат переходить у активний режим, і користувач отримує перше автоматично згенероване повідомлення від системи. Це повідомлення формується Node.js-сервісом як частина початкового сценарію взаємодії та слугує індикатором успішного встановлення з'єднання з

сервером. Поведінка інтерфейсу побудована таким чином, щоб емулювати діалог з оператором, зберігаючи при цьому зрозумілість і простоту взаємодії.

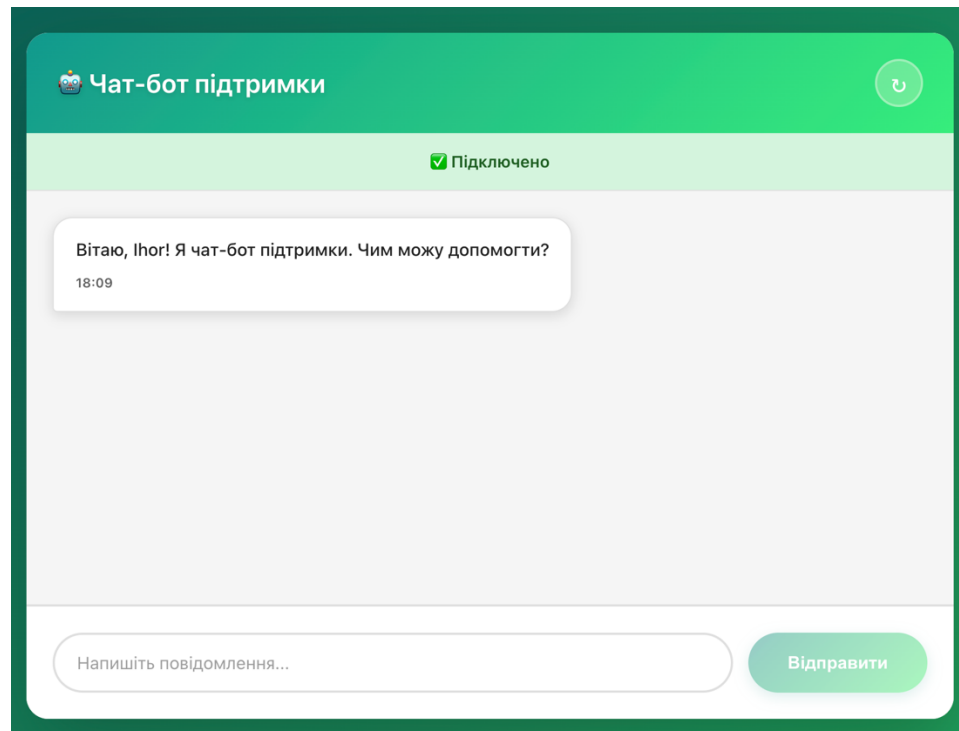


Рис. 3.2. Підключення до сесії

Коли користувач вводить запит, наприклад: «Який статус мого замовлення?», повідомлення негайно передається на Node.js-gateway, який фіксує його у базі даних, визначає статус запиту та паралельно ініціює HTTP-виклик до Python-сервісу / predict_intent. Інтерфейс у цей момент зберігає модель чату в асинхронному режимі, дозволяючи користувачеві продовжувати взаємодію, тоді як серверна частина виконує обчислення.

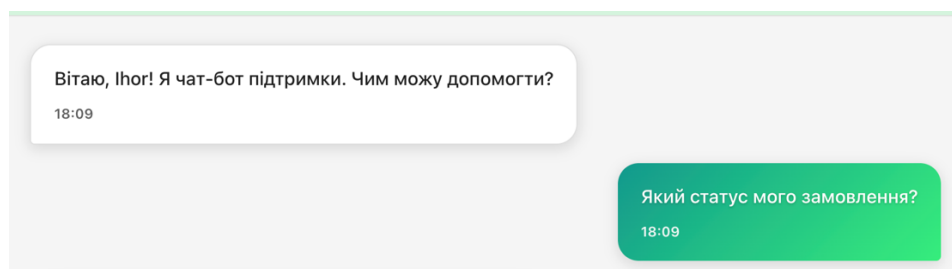


Рис. 3.3. Запит користувача

Після отримання відповіді Python-сервісом починає працювати логіка сценаріїв, визначена у файлі intentScenarios.js. Залежно від ймовірнісної оцінки наміру система може запропонувати користувачеві одразу найбільш імовірний варіант або, якщо впевненість моделі недостатня, — попросити

підтвердити правильний намір з переліку доступних категорій. Такий підхід дозволяє підвищити загальну точність системи та мінімізувати ризик хибної інтерпретації дій користувача.

The screenshot shows a chatbot interface with a green header bar containing a robot icon and the text "Чат-бот підтримки". Below the header is a green bar with a checkmark icon and the text "Підключено". The main content area is white and contains a message from the chatbot: "Схоже, ти хочеш дізнатися статус замовлення 😊" followed by "Намір: перевірити статус замовлення (99%)". Below this is a red link: "Ні, я не бажаю перевірити статус замовлення". The chatbot then asks: "Щоб перевірити статус, поділись з нами інформацією". There are three input fields: "Номер замовлення *" with the value "12345", "Email *" with the value "example@email.com", and "Номер телефону (опціонально)" with the value "+380501234567". At the bottom is a green button labeled "Перевірити статус".

Рис. 3.4. Визначення наміру

The screenshot shows the same chatbot interface as in Figure 3.4, but with a dropdown menu open. The dropdown menu is titled "Оберіть правильний намір:" and contains a list of options: "✓ -- Оберіть --", "Перевірити статус замовлення", "Змінити час доставки", "Скасувати замовлення", "Повернення або відшкодування", "Проблема з оплатою", "Інформація про товар", "Промокод або знижка", "Проблема з акаунтом", "Варіанти доставки", and "Все неправильно, зв'яжіть з оператором". The dropdown menu is positioned over the input fields, partially obscuring them.

Рис. 3.5. Можливість зміни наміру власноруч

Після вибору категорії або підтвердження автоматично визначеного наміру система переходить до формування сценарної відповіді. Відповідь може бути як статичною (коли йдеться про демонстраційні дані), так і динамічною — у випадку інтеграції із зовнішніми сервісами або базами даних. У рамках реалізованого демо-функціоналу система повертає змодельовану інформацію про статус замовлення, після чого пропонує передати запит оператору, що демонструє можливість ескалації та поєднання автоматизованого та ручного обслуговування.

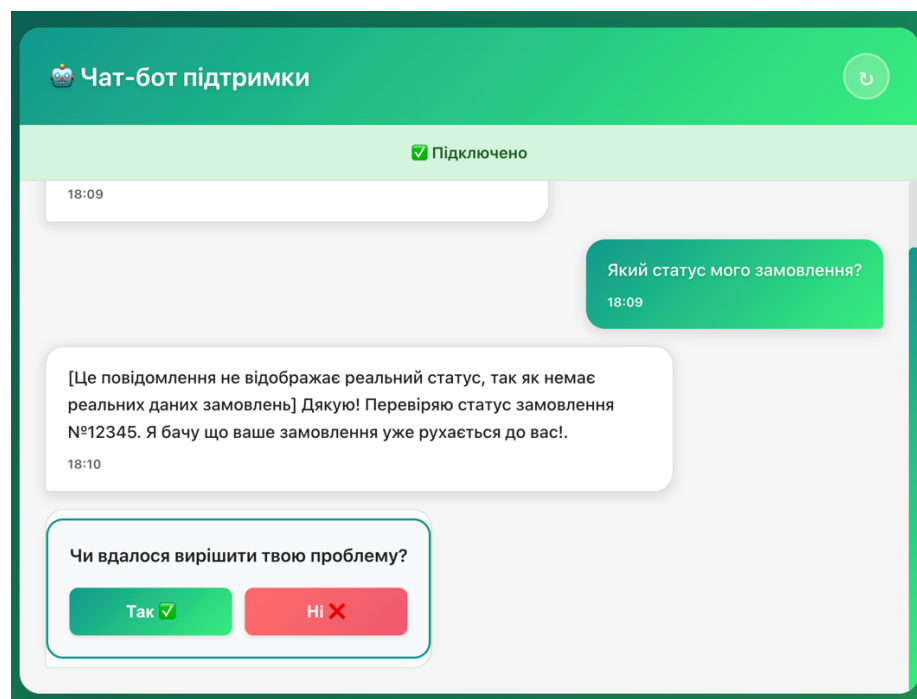


Рис. 3.6. Підтвердження вирішення запиту користувача

Завершальним етапом є передавання користувачеві повідомлення про підключення оператора. Це дає змогу моделі продемонструвати здатність підтримувати повноцінний сценарій обробки запиту, що включає точку передачі діалогу людині, якщо це потрібно. Після того як оператор приєднався до чату, взаємодія між користувачем і оператором продовжується у режимі реального часу завдяки WebSocket-каналу. З боку клієнтського застосунку повідомлення оператора відображаються з відповідною візуальною диференціацією, що допомагає користувачу легко ідентифікувати, з ким він веде діалог. На цьому етапі чат-бот фактично передає ініціативу оператору,

однак зберігає контекст діалогу та продовжує логування всіх подій через gateway-сервіс.

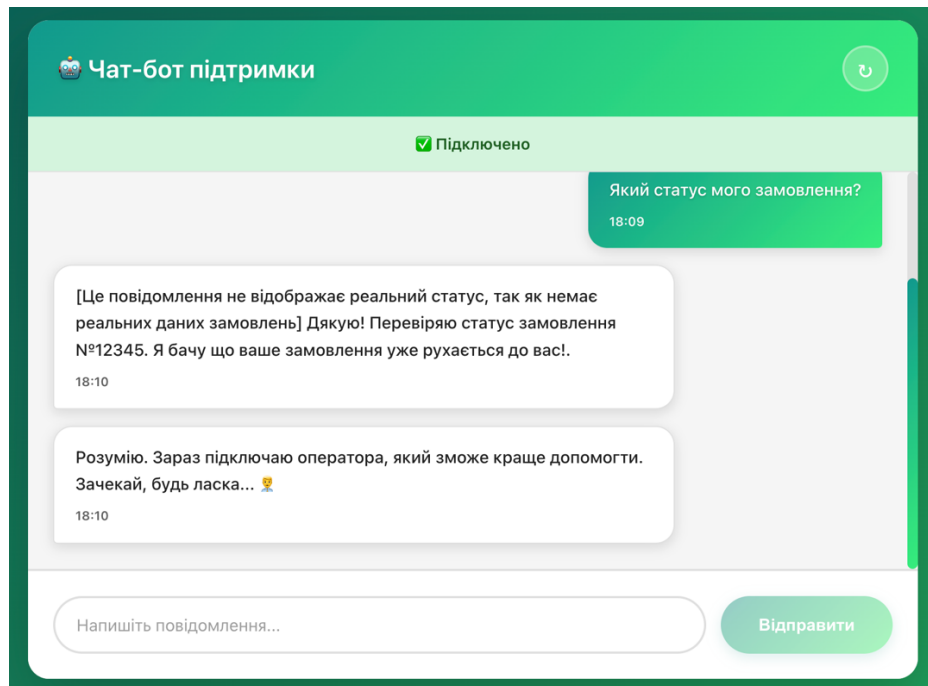


Рис. 3.7. Очікування оператора

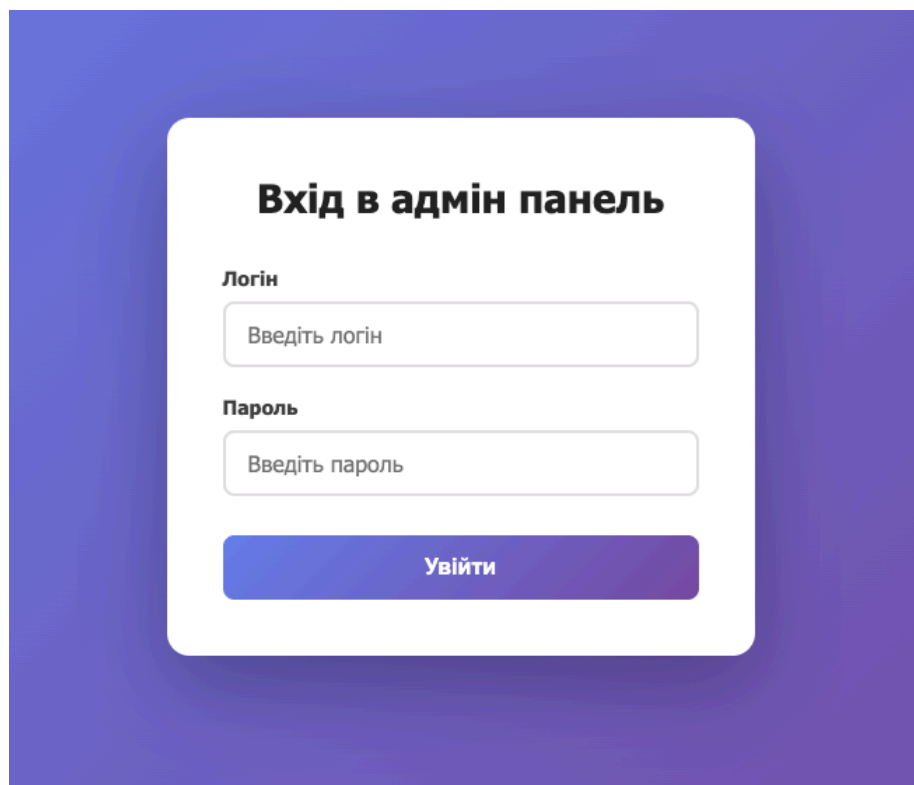


Рис. 3.8. Вхід до адмін панелі оператора

Одночасно в адміністративній панелі оператор отримує повний доступ до інформації щодо активного запиту. Працівник підтримки бачить усю історію

повідомлень, включно з відповідями чат-бота та користувача, а також метадані: ім'я клієнта, номер замовлення, email та телефон (якщо надані). Такий підхід забезпечує достатній контекст для ефективного вирішення проблеми без потреби повторно уточнювати дані. У цьому ж інтерфейсі оператор може завершити діалог, відправити системне повідомлення або передати запит іншому спеціалісту.

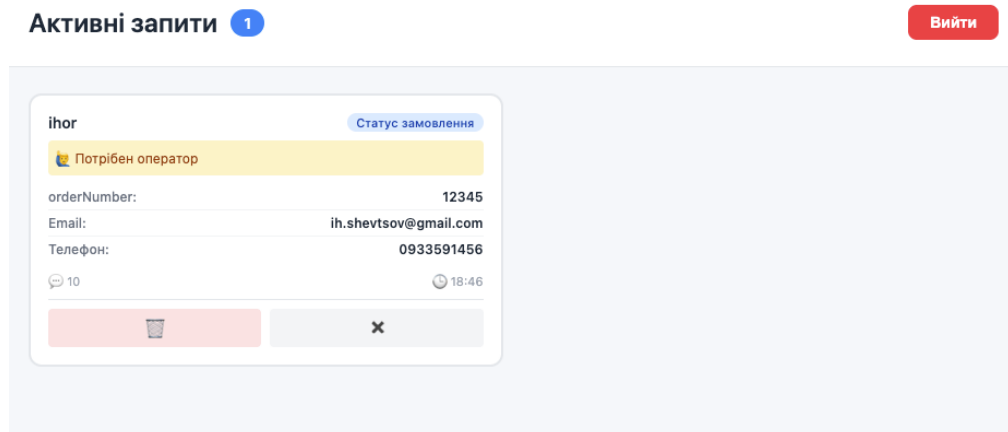


Рис. 3.9. Запити які очікують уваги оператора

Завдяки синхронізації між клієнтським та операторським інтерфейсами кожна дія оператора миттєво відображається в чаті користувача. Наприклад, якщо оператор ставить уточнювальне запитання або підтверджує виконання певної дії, клієнт бачить ці повідомлення з відповідною стилізацією, що сприяє більш природній та комфортній комунікації. Реакції користувача, своєю чергою, миттєво з'являються в інтерфейсі адміністратора, забезпечуючи повноцінний живий діалог. Важливо підкреслити, що інтерфейс адміністратора дозволяє працювати з кількома паралельними чатами. Це особливо актуально для реальних контакт-центрів, де оператор одночасно обслуговує декілька клієнтів. Система відображає кількість активних запитів, а також маркери, що вказують на статус кожного діалогу (наприклад, «очікує відповіді», «приєднано оператора», «вимкнено користувача» тощо). Це підвищує продуктивність і зменшує потребу в додаткових диспетчерських інструментах.

Після завершення діалогу оператор має можливість закрити сесію. У клієнтському застосунку це відображається у вигляді системного

повідомлення, яке вказує, що чат завершено або що оператор завершив обробку запиту. Такий підхід забезпечує чітке інформування користувача та дозволяє уникнути невизначеності у ситуації, коли діалог більше не активний. Завершення сесії також зберігається в базі даних, що дозволяє формувати статистику та аналізувати навантаження.

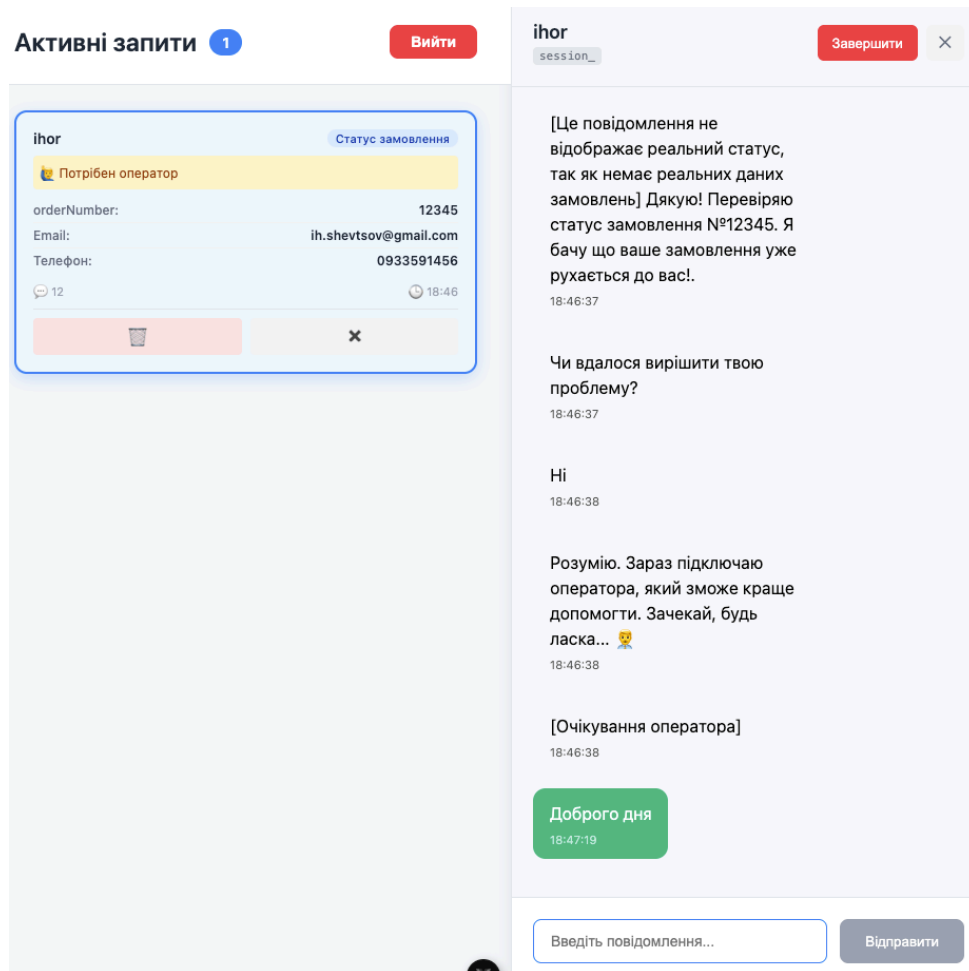


Рис. 3.10. Інтерфейс оператора під час спілкування з користувачем

Висновки до розділу 3

У третьому розділі було узагальнено результати практичної реалізації чатботу підтримки, побудованого на основі моделі визначення намірів користувачів. У межах цього розділу розгорнуто продемонстровано, яким чином теоретичні та алгоритмічні напрацювання, описані у попередніх розділах, були інтегровані у повноцінну програмну систему, що складається з клієнтського застосунку, мікросервісної інфраструктури та адміністративної панелі для операторів. На відміну від попередніх розділів, де основна увага

була зосереджена на підходах до моделювання та оцінці точності алгоритмічної частини, цей розділ описує практичні аспекти побудови рішення, ілюструючи взаємодію між компонентами системи на реальних прикладах.

У результаті виконаної роботи було визначено та формалізовано вимоги до функціональних можливостей системи, включно з необхідністю обробки природної мови, безперервної взаємодії із користувачем у режимі реального часу, врахуванням можливості підключення оператора та забезпеченням відповідного рівня стійкості до помилок. Було обґрунтовано вибір інструментальних засобів, що охоплюють стек технологій Python для виконання інференсу моделі XLM-RoBERTa, Node.js для управління бізнес-логікою та каналами WebSocket, MongoDB як базу даних для збереження історії запитів, а також Vue 3 — як основу для клієнтського інтерфейсу.

Особливу увагу приділено проєктуванню та реалізації окремих модулів системи. У роботі детально описано інтеграцію моделі визначення намірів у Python-сервіс, механізми маршрутизації запитів через gateway-компонент, принципи роботи адміністративної панелі та особливості взаємодії між клієнтом і оператором у межах спільної сесії. Наведені приклади роботи системи демонструють, що взаємодія між компонентами відбувається коректно, а сам чатбот здатний не лише інтерпретувати запит користувача, але й забезпечувати плавний перехід до взаємодії з оператором у разі потреби. Це свідчить про готовність архітектури до масштабування і подальшого розширення функціоналу.

Таким чином, у третьому розділі було підтверджено можливість побудови працездатної системи підтримки користувачів, яка поєднує переваги сучасних NLP-технологій і гнучких принципів мікросервісної архітектури. Реалізоване рішення демонструє, що інтеграція моделі визначення намірів у реальний програмний продукт дозволяє суттєво оптимізувати первинну обробку звернень, скоротити навантаження на операторів і створити фундамент для подальшого удосконалення чатботів у сервісних системах.

ВИСНОВКИ

У ході виконання магістерської роботи було здійснено комплексне дослідження проблематики оптимізації ресурсів контакт-центрів шляхом упровадження системи автоматичного визначення намірів користувачів на основі методів обробки природної мови та нейронних мереж. У роботі проаналізовано сучасний стан ринку контакт-центрів, визначено ключові виклики їх функціонування, серед яких істотне зростання обсягів звернень, збільшення вартості обслуговування, потреба в оперативності відповіді та необхідність зменшення навантаження на операторів. Проведений огляд наявних технологій продемонстрував, що традиційні підходи на основі IVR-систем та ручних сценаріїв дедалі гірше відповідають сучасним вимогам, тоді як автоматизовані рішення на основі NLP і машинного навчання забезпечують значно вищу точність, масштабованість і стабільність роботи.

У дослідженні обґрунтовано доцільність використання трансформерних архітектур для задачі класифікації намірів. Було підготовлено власний доменний датасет, сформований з урахуванням особливостей україномовних звернень користувачів. Задіяно два підходи для моделювання: базову модель із застосуванням TF-IDF та логістичної регресії, а також високорівневу трансформерну модель XLM-RoBERTa, що показала значно вищу стійкість до мовних варіацій та семантичних відхилень. Порівняльний аналіз моделей продемонстрував перевагу трансформера, зокрема у складних класах, де базові методи втрачали точність. Отримані експериментальні результати підтвердили, що модель XLM-RoBERTa є ефективним рішенням для реальної інтеграції в чат-боти підтримки.

Окрему увагу у роботі приділено розробці повноцінної системи чат-боту, яка поєднує модель визначення намірів із мікросервісною архітектурою. Спроектовано та реалізовано декілька ключових модулів: сервіс машинного навчання, що надає API для класифікації намірів; Node.js-gateway, який відповідає за взаємодію з веб-клієнтом, маршрутизацію запитів, запис подій у базу даних та управління WebSocket-комунікацією; адміністративну панель,

що забезпечує моніторинг активних сесій та можливість втручання оператора. Створений прототип продемонстрував коректну роботу як у локальному середовищі, так і в інфраструктурній моделі, орієнтованій на розгортання в AWS. З урахуванням вимог до масштабованості було показано, що мікросервісний підхід дозволяє гнучко оновлювати модель, додавати нові сценарії та адаптувати систему до змін у навантаженні.

У результаті реалізації чат-боту підтверджено, що впровадження автоматичного визначення намірів здатне суттєво знизити навантаження на операторів, забезпечити швидшу реакцію на запити користувачів та підвищити якість обслуговування без пропорційного зростання людських ресурсів. Експериментальна частина роботи показала, що система демонструє високу точність у реальних сценаріях і здатна не лише відповідати на типові запити, а й забезпечувати адаптивний перехід до оператора у випадках, що потребують додаткової обробки.

Підсумовуючи виконану роботу, можна стверджувати, що поставлена мета досягнута в повному обсязі: проведено аналіз предметної області, сформовано вимоги до системи, розроблено й апробовано алгоритмічне та програмне забезпечення чат-боту підтримки. Запропонований підхід продемонстрував практичну цінність і може слугувати основою для подальшого розвитку систем контакт-центрів, зокрема шляхом розширення набору намірів, інтеграції зовнішніх API для доступу до даних замовлень, удосконалення діалогових стратегій і впровадження контекстних моделей, здатних підтримувати багатокрокові діалоги. Отримані результати мають потенціал для подальшої масштабної інтеграції в реальні бізнес-процеси та можуть бути використані як база для наступних досліджень у сфері інтелектуальних агентів та автоматизації клієнтської підтримки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Treglown L, Furnham A. Predicting Performance of Call Center Staff: The Role of Cognitive Ability and Emotional Intelligence. *Psychology International*. 2024; 6(4):903-916. <https://doi.org/10.3390/psycholint6040058>
2. Altarif, B., Al Mubarak, M. (2022). Artificial Intelligence: Chatbot—The New Generation of Communication. In: Hamdan, A., Harraf, A., Arora, P., Alareeni, B., Khamis Hamdan, R. (eds) *Future of Organizations and Work After the 4th Industrial Revolution*. Studies in Computational Intelligence, vol 1037. Springer, Cham. https://doi.org/10.1007/978-3-030-99000-8_12
3. Mangipudi, Phanidra, AI Powered Chatbots Vs Human Agents, A Compatible Study On Customer Satisfaction (March 08, 2025). <http://dx.doi.org/10.2139/ssrn.5176219>
4. SelectVoice Communications. Future of Call Centers: 30 Must-Know Statistics and Trends [Електронний ресурс] – Режим доступу до ресурсу: <https://selectvoicecom.com/content-articles/future-of-call-centers-30-must-know-statistics-and-trends>.
5. Chicu, Dorina and Ryan, Gerard and Mirela, Valverde, Determinants of Customer Satisfaction in Call Centres February 12, 2016. *European Accounting and Management Review*, 2016, 2(2), 20-41., Available at SSRN: <https://ssrn.com/abstract=2888793>
6. Nasution, Maulana & Arianto, Arianto. (2024). Exploring the Role of AI Chatbot in English Language Teaching and Learning. *Journal of English Language and Education*. 9. 34-43. <https://doi.org/10.31004/jele.v9i5.513>.
7. Evgeny Tebenkov, Igor Prokhorov, Machine learning algorithms for teaching AI chat bots, *Procedia Computer Science*, Volume 190, 2021, Pages 735-744, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2021.06.086>.
8. Aldhafeeri, Lama & Aljumah, Fay & Thabyan, Fajr & Alabbad, Maram & AlShahrani, Sultanh & Alanazi, Fawzia & Alnafjan, Abeer. (2025). Generative AI Chatbots Across Domains: A Systematic Review. *Applied Sciences*. 15. 11220. <https://doi.org/10.3390/app152011220>.

9. Genesys Cloud CX [Электронный ресурс] – Режим доступа до ресурсу: <https://www.genesys.com/genesys-cloud>
10. Five9 AI Agent Assist [Электронный ресурс] – Режим доступа до ресурсу: <https://www.five9.com/products/capabilities/agent-assist>
11. Amazon Connect [Электронный ресурс] – Режим доступа до ресурсу: <https://aws.amazon.com/connect/>
12. Amazon Lex - AI Chat Builder [Электронный ресурс] – Режим доступа до ресурсу: <https://aws.amazon.com/lex>
13. Amazon Transcribe [Электронный ресурс] – Режим доступа до ресурсу: <https://aws.amazon.com/transcribe>
14. Conversational Agents (Dialogflow CX) [Электронный ресурс] – Режим доступа до ресурсу: <https://docs.cloud.google.com/dialogflow/cx/docs>
15. Microsoft Bot Framework [Электронный ресурс] – Режим доступа до ресурсу: <https://learn.microsoft.com/ru-ru/azure/bot-service/bot-builder-basics?view=azure-bot-service-4.0>
16. Surdeanu M, Valenzuela-Escárcega MA. Using Transformers with the Hugging Face Library. In: Deep Learning for Natural Language Processing: A Gentle Introduction. Cambridge University Press. 2024. 194-215. <https://doi.org/10.1017/9781009026222.014>.
17. Masaling, Nikita & Suhartono, Derwin. (2024). Utilizing RoBERTa and XLM-RoBERTa pre-trained model for structured sentiment analysis. International Journal of Informatics and Communication Technology (IJ-ICT). 13. 410. <https://doi.org/10.11591/ijict.v13i3.pp410-421>.
18. Auger, T., Saroyan, E. Overview of the OpenAI APIs. Generative AI for Web Development (book chapter). Apress, Berkeley, CA. 2024. https://doi.org/10.1007/979-8-8688-0885-2_6.
19. Santoso, Gerald & Setiawan, Johan & Sulaiman, Agus. (2023). Development of OpenAI API Based Chatbot to Improve User Interaction on the JBMS Website. G-Tech: Jurnal Teknologi Terapan. 7. 1606-1615. <https://doi.org/10.33379/gtech.v7i4.3301>.

20. Documentation for the JSON Lines text file format [Электронный ресурс] – Режим доступа до ресурсу: <https://jsonlines.org>
21. Matthew J. Lavin. Analyzing Documents with TF-IDF. The Programming Historian. Issue 8. 2019. <https://doi.org/10.46430/phen0082>.
22. Javed, Awais & Rashid, Imran & Tahir, Shahzaib & Saeed, Saqib & Almuhaideb, Abdullah & Alissa, Khalid. (2024). AdamW+: Machine Learning Framework to Detect Domain Generation Algorithms for Malware. IEEE Access. PP. 1-1. <https://doi.org/10.1109/ACCESS.2024.3407546>.
23. Vue 3 documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://vuejs.org/guide/introduction>
24. WebSockets documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>
25. Mongodb documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://www.mongodb.com>
26. Fast API documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://fastapi.tiangolo.com/reference>

ДОДАТКИ

```
function buildPrompt({ intent, seedPhrases, slotSchema, useSurzhyk, lengthHint,
recentExamples, tone, slotProfile }) {
  const recentBlock = recentExamples.length
    ? `Приклади фраз, які ВЖЕ були згенеровані раніше (їх НЕ можна повторювати та
робити надто подібними за формулюванням):

${recentExamples.map((s) => `– ${s}`).join("\n")}`
    : '';

  const surzhykHint = useSurzhyk
    ? `МОВА: СУРЖИК
– Можна змішувати українські та російські слова ("заказ", "жду", "доставка",
"трекінг", "посилка").
– Дозволені розмовні конструкції, неідеальна граматики, спрощення.`
    : `МОВА: РОЗМОВНА УКРАЇНСЬКА
– Без явного суржику, але можна використовувати прості розмовні конструкції.`;

  const slotsDescription = Object.keys(slotSchema).length
    ? `СЛОТИ КОНТЕКСТУ ДЛЯ intent "${intent}":

${Object.entries(slotSchema)
  .map(([name, desc]) => `– ${name}: ${desc}`)
  .join("\n")}`
    : `Для intent "${intent}" слоти контексту не використовуються, тому "slots" =
{}.`;

  const styleHints = `
ДОДАТКОВІ ВИМОГИ ДО СТИЛЮ:
– Це живе повідомлення в чаті підтримки, а не офіційний лист.
– Можна пропускати деякі коми, використовувати повтори ("дуже-дуже чекаю").
– Частина фраз роби короткими й різкими: "Де моє замовлення 1234?", "Коли буде
доставка?".
– Додатково можна додати час/контекст:
  "зранку", "ввечері", "сьогодні", "вчора ввечері", "на вихідних", "до вечора", "після
роботи" тощо.

ФОРМАТ НОМЕРА ЗАМОВЛЕННЯ (якщо він є):
– Не завжди використовуй формат "№1234".
– Змішуй:
  "замовлення №1234", "замовлення 1234", "номер замовлення 1234", "номер 1234", "заказ
1234", чи просто "1234" у контексті.
`;

  return `
Ти генеруєш клієнтські запити для служби підтримки інтернет-магазину українською
мовою.
```

Intent: "\${intent}".

Приклади фраз (seed), на які можна орієнтуватися за змістом, але не копіювати:

```
${seedPhrases.map((s) => `– ${s}`).join("\n")}
```

```
${recentBlock}
```

```
${surzhykHint}
```

```
${toneHint(tone)}
```

```
${slotProfileHint(slotProfile)}
```

```
${styleHints}
```

```
${slotsDescription}
```

ЗАВДАННЯ:

- Згенеруй ОДИН приклад клієнтського повідомлення, яке відповідає intent "\${intent}".
- Фраза має звучати природно, як повідомлення реального клієнта в чаті.
- Уникай повтору однакових шаблонів (особливо однакового початку).
- \${lengthHint}

ФОРМАТ ВИВОДУ:

- Поверни ТІЛЬКИ ВАЛІДНИЙ JSON-ОБ'ЄКТ:

```
{
  "text": "...",
  "intent": "${intent}",
  "slots": {
    "...": "..."
  }
}
```

- "intent" завжди "\${intent}".
 - "slots" – ключі згідно опису слотів.
 - Якщо для певного слоту немає явного значення в тексті – встанови його як null.
 - Не додавай жодних пояснень, коментарів чи markdown поза JSON.
- ```
`;
}
```

```

prepare_intent_dataset.py
import json
from pathlib import Path
from sklearn.model_selection import train_test_split

data_dir = Path("../dataset")
rows = []

for path in data_dir.glob("dataset_*.jsonl"):
 with open(path, "r", encoding="utf-8") as f:
 for line in f:
 obj = json.loads(line)
 rows.append({
 "text": obj["text"],
 "intent": obj["intent"],
 })

print(f"Всього прикладів: {len(rows)}")

train/val/test
train_val, test = train_test_split(rows, test_size=0.1, random_state=42,
stratify=[r["intent"] for r in rows])
train, val = train_test_split(train_val, test_size=0.1, random_state=42,
stratify=[r["intent"] for r in train_val])

def save_jsonl(path, items):
 with open(path, "w", encoding="utf-8") as f:
 for obj in items:
 f.write(json.dumps(obj, ensure_ascii=False) + "\n")

out_dir = Path("./intent_data")
out_dir.mkdir(exist_ok=True)

save_jsonl(out_dir / "train.jsonl", train)
save_jsonl(out_dir / "val.jsonl", val)
save_jsonl(out_dir / "test.jsonl", test)

print("Зроблено: intent_data/train.jsonl, val.jsonl, test.jsonl")

```

```

train_intent_baseline.py
import json
from pathlib import Path
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report

def load_jsonl(path):
 items = []
 with open(path, "r", encoding="utf-8") as f:
 for line in f:
 items.append(json.loads(line))
 return items

data_dir = Path("./intent_data")
train = load_jsonl(data_dir / "train.jsonl")
val = load_jsonl(data_dir / "val.jsonl")
test = load_jsonl(data_dir / "test.jsonl")

X_train = [r["text"] for r in train]
y_train = [r["intent"] for r in train]

X_val = [r["text"] for r in val]
y_val = [r["intent"] for r in val]

X_test = [r["text"] for r in test]
y_test = [r["intent"] for r in test]

model = Pipeline([
 ("tfidf", TfidfVectorizer(
 analyzer="word",
 ngram_range=(1, 2),
 min_df=2
)),
 ("clf", LogisticRegression(max_iter=1000))
])

model.fit(X_train, y_train)

print("=== VAL ===")
y_pred_val = model.predict(X_val)
print(classification_report(y_val, y_pred_val))

print("=== TEST ===")
y_pred_test = model.predict(X_test)
print(classification_report(y_test, y_pred_test))

import joblib
joblib.dump(model, "intent_baseline.joblib")

```

```

train_intent_transformer.py
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
TrainingArguments, Trainer
import json

dataset = load_dataset(
 "json",
 data_files={
 "train": "intent_data/train.jsonl",
 "validation": "intent_data/val.jsonl",
 "test": "intent_data/test.jsonl",
 }
)

intents = sorted(list({row["intent"] for row in dataset["train"]}))
label2id = {label: i for i, label in enumerate(intents)}
id2label = {i: label for label, i in label2id.items()}

def encode_labels(example):
 example["label"] = label2id[example["intent"]]
 return example

dataset = dataset.map(encode_labels)

model_name = "xlm-roberta-base"
tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize_batch(batch):
 return tokenizer(
 batch["text"],
 padding="max_length",
 truncation=True,
 max_length=64,
)

tokenized = dataset.map(tokenize_batch, batched=True)

model = AutoModelForSequenceClassification.from_pretrained(
 model_name,
 num_labels=len(intents),
 id2label=id2label,
 label2id=label2id,
)

args = TrainingArguments(
 output_dir="./intent_xlmr",
 eval_strategy="epoch",
 save_strategy="epoch",
 learning_rate=2e-5,
 per_device_train_batch_size=16,
 per_device_eval_batch_size=32,

```

```

 num_train_epochs=3,
 weight_decay=0.01,
)

 def compute_metrics(eval_pred):
 logits, labels = eval_pred
 preds = logits.argmax(axis=-1)
 from sklearn.metrics import accuracy_score, f1_score
 return {
 "accuracy": accuracy_score(labels, preds),
 "f1": f1_score(labels, preds, average="weighted"),
 }

 trainer = Trainer(
 model=model,
 args=args,
 train_dataset=tokenized["train"],
 eval_dataset=tokenized["validation"],
 compute_metrics=compute_metrics,
)

 trainer.train()
 trainer.save_model("./intent_xlmr")
 tokenizer.save_pretrained("./intent_xlmr")

```