Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)
Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів класифікації текстових повідомлень в діалогах
(тема)

Виконав:
здобувач 2 року навчання
групи ІПЗм-23-3
Олександр ЛОГВІНОВ
(власне ім'я, прізвище)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Олексій ТУРУТА
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Логвінову Олександру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів класифікації текстових повідомлень в діалогах»
   затверджена наказом університету від 15 квітня 2025 р. № 290Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.
   Вихідні дані роботи: календарний план роботи, методичні вказівки до оформлення пояснювальної записки, перелік методів навчання для класифікації текстових повідомлень
3. Перелік питань, що потрібно опрацювати у роботі: вступ; аналіз предметної галузі, огляд існуючих підходів, їхніх обмежень і сучасних тенденцій; аналіз наукових і літературних джерел; формулювання задачі; теоретичне дослідження статистичних мовних моделей, нейронних мережевих підходів, попередньо навчених моделей і великих мовних моделей; порівняльний аналіз підходів; інтерпретація отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	16.04.2025	Виконано
2	Аналіз предметної галузі і постановка задачі	17.04.2025	Виконано
3	Теоретичне дослідження	19.04.2025	Виконано
4	Практичне дослідження	28.04.2025	Виконано
5	Підготовка до апробації результатів дослідження. Публікація матеріалів	01.05.2025	Виконано
6	Підготовка пояснювальної записки	08.05.2025	Виконано
7	Підготовка презентації та доповіді	20.05.2025	Виконано
8	Перевірка на плагіат	01.06.2025	Виконано
9	Нормоконтроль	03.06.2025	Виконано
10	Рецензування	04.06.2025	Виконано
11	Попередній захист	08.06.2025	Виконано
12	Занесення диплома в електронний архів	09.06.2025	Виконано
13	Допуск до захисту у зав. кафедри	10.06.2025	Виконано

Дата видачі завдання 16 квітня 2025р.

Здобувач _____ Олександр ЛОГВІНОВ
(підпис)

Керівник роботи _____ доц. Олексій ТУРУТА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 75 с., 6 рис., 10 табл., 10 джерел

АВТОМАТИЗАЦІЯ ЧАТІВ, АНАЛІЗ МЕТАДАНИХ, ГЛИБОКЕ НАВЧАННЯ, ДІАЛОГИ, КЛАСИФІКАЦІЯ ТЕКСТУ, КЛАСТЕРІЗАЦІЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, МАШИННЕ НАВЧАННЯ, BERT

Об’єктом дослідження є текстові повідомлення у діалогах та методи їх автоматизованої класифікації.

Метою дослідження є аналіз продуктивності та точності різних методів класифікації текстових повідомлень у діалогах. Зокрема, порівнюються статистичні підходи, моделі на основі трансформерів (зокрема BERT), Сховані Марковські Моделі (HMM), модифікована версія алгоритму UFTR (Unified Framework for Ticket Routing), мультимодальний підхід із урахуванням медіа-даних, що супроводжують текст, а також наївний байєсівський підхід

Основними методами дослідження були аналіз архітектур обробки природної мови та експериментальне порівняння отриманих результатів за допомогою стандартних метрик (точність, повнота, F1-міра). З метою підвищення ефективності класифікації для кожного підходу підбиралися індивідуальні параметри, аби знайти оптимальні значення.

У результаті дослідження було створено дві статистичні моделі (HMM та Bayes), а також дві моделі на базі трансформерів, а саме BERT і CLIP. Остання модель враховує не лише текстову складову, але й мультимедійні дані, які часто супроводжують текстове повідомлення.

Аналіз отриманих результатів продемонстрував переваги використання трансформер-архітектур у задачах обробки природної мови, особливо для оцінювання точності класифікації. Крім того, інтеграція меді даних дозволяє суттєво поліпшити ефективність класифікації у випадках складних або неоднозначних текстів.

CHAT AUTOMATION, METADATA ANALYSIS, DEEP LEARNING, DIALOGUES, TEXT CLASSIFICATION, CLUSTERING, NATURAL LANGUAGE PROCESSING MACHINE LEARNING, BERT

The object of study is text messages in dialogs and methods of their automated classification.

The purpose of the study is to analyze the performance and accuracy of various methods for classifying text messages in dialogues. In particular, we compare statistical approaches, transformer-based models (including BERT), Hidden Markov Models (HMM), a modified version of the UFTR (Unified Framework for Ticket Routing) algorithm, a multimodal approach based on media data accompanying the text, and a naive Bayesian approach.

The main research methods were the analysis of natural language processing architectures and the experimental comparison of the results obtained using standard metrics (accuracy, completeness, F1-measure). To improve the classification efficiency, individual parameters were selected for each approach to find the optimal values.

The study resulted in the creation of two statistical models (HMM and Bayes), as well as two transformer-based models, namely BERT and CLIP. The latter model takes into account not only the text component but also multimedia data that often accompanies a text message.

The analysis of the obtained results demonstrated the advantages of using transformer architectures in natural language processing tasks, especially for assessing classification accuracy. In addition, the integration of media data can significantly improve classification efficiency in cases of complex or ambiguous texts.

Завідувачу кафедри
ПІ_____
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ_
(вчене звання, сласне ім'я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я, Логвінов Олександр Володимирович_____
(прізвище, ім'я, по батькові)

здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи___ІПЗм-23-3_____

кафедра_____ програмної інженерії _____,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему __« Дослідження методів_____ класифікації текстових повідомлень в діалогах»_____ ___,
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в репозиторії "ElArKhNURE". Погоджуюся з авторським договором, відповідно до Положення про репозиторій ХНУРЕ "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата Підпис

ЗМІСТ

ВСТУП

Сучасні технології обробки природної мови (NLP) відіграють ключову роль у багатьох сферах, зокрема в автоматизації класифікації текстових даних. Зростаюча кількість цифрових комунікацій, таких як чати, електронна пошта та соціальні мережі, створює необхідність ефективних автоматизованих підходів до аналізу та класифікації текстових повідомлень. У цьому контексті задачі, пов’язані з класифікацією тексту в діалогах, стають особливо актуальними, оскільки вони включають не лише аналіз змісту, але й врахування структурованих даних, таких як метадані (автор, час відправлення, категорія теми тощо).

Розвиток моделей на основі трансформерів, таких як BERT, значно підвищив точність обробки текстових даних, зокрема завдяки врахуванню контексту слів і речень. Одночасно, ймовірнісні моделі, наприклад, Сховані Маркові Моделі (HMM), залишаються популярними для задач, де важливі послідовності даних. Крім того, новітні підходи, такі як UFTR (Unified Framework for Ticket Routing), демонструють ефективність у врахуванні взаємозв’язків між текстовими даними та їх структурованими характеристиками.

Актуальність теми полягає в необхідності розробки методів, які забезпечують високу точність і ефективність класифікації текстових повідомлень у діалогах. Це особливо важливо для автоматизації роботи чатів, маршрутизації звернень у системах підтримки клієнтів та аналізу настроїв у соціальних мережах. Дослідження спрямоване на вирішення проблем недостатньої точності при роботі з неоднозначними текстами або текстами з багатими метаданими.

Мета роботи полягає у створенні ефективних моделей класифікації текстових повідомлень у діалогах шляхом порівняння підходів на основі трансформера BERT, алгоритму Вітербі для HMM та адаптованої версії UFTR.

Завдання роботи включають аналіз сучасних архітектур NLP для класифікації текстових повідомлень та розробку трьох моделей: на основі BERT, алгоритму Вітербі для HMM та адаптованого UFTR.

Об’єктом дослідження є текстові повідомлення у діалогах та методи їх автоматизованої класифікації.

Предметом дослідження є методи класифікації тексту, що використовують сучасні архітектури обробки природної мови та враховують взаємозв’язки метаданих.

Методи дослідження включають використання трансформерів, зокрема моделі BERT, для аналізу текстових даних, що дозволяє враховувати контекстуальну інформацію слів та речень. Для аналізу послідовностей даних застосовуються ймовірнісні підходи на основі Схованих Маркових Моделей (HMM), які ефективно працюють з текстами, що мають чітку послідовність. Крім того, адаптація алгоритму UFTR забезпечує врахування структурованих характеристик тексту, таких як час відправлення, авторство чи тематична категорія, що дозволяє моделі більш точно і повно класифікувати повідомлення. Для оцінки продуктивності розроблених моделей використовуються стандартні метрики класифікації, такі як точність, повнота та F1-міра, що забезпечує об'єктивне порівняння результатів.

Одержані результати демонструють ефективність сучасних архітектур NLP у задачах класифікації текстових повідомлень. Розроблені моделі досягають високої точності, а їх адаптація для інтеграції метаданих суттєво покращує результати для складних текстів. Застосування результатів можливе у системах підтримки клієнтів, автоматизації діалогів та аналізі соціальних мереж.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Тема дослідження – класифікація текстових повідомлень у діалогах – належить до галузі обробки природної мови (Natural Language Processing, NLP), яка активно розвивається у зв'язку з ростом цифрових комунікацій. Це включає чати, електронну пошту, соціальні мережі та системи підтримки клієнтів, які генерують значний обсяг текстових даних. Ефективна класифікація таких даних є критично важливою для підвищення автоматизації, зменшення витрат і поліпшення якості сервісів.

1.1 Моделі трансформерів

Моделі трансформерів є сучасними інструментами в галузі обробки природної мови (NLP), які показують вражаючі результати у багатьох задачах, включаючи класифікацію тексту. Їх основою є механізм самоуваги (self-attention), що дозволяє аналізувати відносини між словами в тексті незалежно від їх розташування. Ця особливість дає змогу трансформерам враховувати широкий контекст, що робить їх особливо ефективними для задач, де значення слова залежить від його оточення.

Однією з найвідоміших моделей є BERT (Bidirectional Encoder Representations from Transformers). Вона відрізняється двонаправленим аналізом, що дозволяє враховувати контекст як зліва, так і справа від слова. Це робить її надзвичайно точною для задач класифікації тексту, таких як аналіз настроїв або категоризація тем. Під час тренування BERT використовує два основні підходи: передбачення замаскованих слів у тексті (Masked Language Model) і визначення логічності продовження речень (Next Sentence Prediction). Такий підхід допомагає моделі глибше розуміти текст. Однак висока обчислювальна складність і потреба у великих ресурсах є її значними обмеженнями.

Інша популярна модель, GPT (Generative Pre-trained Transformer), відрізняється однонаправленим контекстом, тобто аналізує текст зліва направо. Це робить її дуже ефективною для генерації текстів, але менш придатною для задач, де важливий повний контекст. GPT тренується на завданні передбачення

наступного слова, що дозволяє їй генерувати тексти високої якості. Хоча ця модель простіша у використанні, її обмеженням є складність у врахуванні багатозначних текстів.

Трансформери показують значну ефективність у задачах класифікації тексту завдяки їх здатності глибоко аналізувати контекст. Вони використовуються для попереднього навчання на великих корпусах даних, а потім донавчаються на спеціалізованих задачах, таких як аналіз настроїв або класифікація за темами. Завдяки додатковому класифікаційному шару ці моделі можуть бути адаптовані до різноманітних задач NLP.

1.2 Ймовірнісні підходи (HMM)

Сховані Марковські моделі (HMM, Hidden Markov Models) є інструментом для моделювання послідовностей даних, які мають латентну структуру. Ці моделі використовуються в задачах озпізнавання мовлення, аналіз текстових повідомлень та класифікація тексту.

Сховані Марковські моделі базуються на ймовірнісному підході до аналізу послідовностей, де спостереження є видимими даними, такими як слова або символи тексту, а сховані стани виступають невидимими класами чи категоріями, які пояснюють генерацію цих спостережень. Перехідні ймовірності визначають динаміку змін між схованими станами, а ймовірності спостережень оцінюють, наскільки певне спостереження ймовірно в конкретному стані. HMM дозволяє моделювати залежності між елементами послідовності, використовуючи обмежену кількість параметрів, що робить їх ефективним інструментом для аналізу структурованих даних.

Метою використання HMM є знаходження оптимальної послідовності схованих станів, які найкраще пояснюють спостережувані дані. Для реалізації HMM використовуються три основні алгоритми. Алгоритм вперед-назад (Forward-Backward) застосовується для обчислення ймовірності спостережень і латентних станів, забезпечуючи ефективне оцінювання розподілу ймовірностей по всій послідовності. Алгоритм Вітербі забезпечує знаходження найбільш

імовірної послідовності схованих станів, що особливо корисно для задач розпізнавання послідовностей. Алгоритм Баума-Велша реалізує оцінку параметрів HMM за допомогою очікувально-максимізаційного підходу (EM-алгоритм), дозволяючи оптимізувати моделі на основі неповних даних.

У діалогах HMM може використовуватися для визначення теми або категорії повідомлень шляхом аналізу послідовностей слів, які відповідають певним темам або категоріям. Наприклад, HMM може моделювати ймовірності переходів між словами чи фразами, що дозволяє виявляти шаблони, притаманні конкретним темам діалогу.

1.3 Мультимодальні підходи

У діалогах користувачів текстові повідомлення часто супроводжуються фотографіями та відео. Для ефективного аналізу таких повідомлень застосовується мультимодальний підхід, який поєднує інформацію з різних джерел, таких як текст, зображення, аудіо чи метадані. Під час аналізу враховуються час створення повідомлення, дані про автора, платформа, а також контекст розмови.

Для реалізації таких систем використовуються трансформери, наприклад Vision-and-Language Transformers (ViLT), які інтегрують текстові та візуальні дані для створення спільних репрезентацій. Для вилучення ознак застосовуються Convolutional Neural Networks (CNN), які аналізують зображення, у поєднанні з Recurrent Neural Networks (RNN) або трансформерами для текстових даних.

Використання метаінформації, такої як час створення повідомлення чи інформація про автора, дозволяє зменшити неоднозначності під час класифікації. Інтеграція тексту з іншими джерелами, наприклад, емодзі або зображеннями, допомагає точніше оцінити емоційний контекст повідомлення.

У підсумку, мультимодальні підходи дають змогу враховувати ширший контекст завдяки використанню різних модальностей, підвищують точність аналізу та роблять результати стійкішими до неоднозначностей тексту. Такі

підходи добре працюють у складних сценаріях, де текст супроводжується іншими типами даних.

1.4 Алгоритм UFTR

Unified Framework for Ticket Routing (UFTR) є алгоритмом, який був розроблений для вирішення задач маршрутизації повідомлень або запитів у великих системах підтримки клієнтів. У контексті класифікації текстових повідомлень UFTR може інтегрувати текстову інформацію запиту, метаінформацію, включаючи час створення чи джерело повідомлення, а також інформацію про автора або платформу, з якої було надіслано текст. Це дозволяє створювати більш точні та адаптивні рішення для класифікації.

Алгоритм також враховує історію класифікації та маршрутизації повідомлень, аналізуючи, які категорії або групи вже обробляли подібні задачі. Для цього використовуються графи або нейронні мережі, які моделюють залежності між повідомленнями та можливими класами. Цей підхід дозволяє підвищити ефективність класифікації завдяки урахуванню попередніх рішень, забезпечуючи точніше зіставлення тексту до відповідної категорії.

UFTR працює як модель ранжування, що оцінює кожну пару "повідомлення-категорія" на основі набору ознак і обчислює рейтинг для кожного можливого класу. Під час класифікації текстових повідомлень це дозволяє врахувати специфіку контексту, ключові слова тексту, а також додаткові характеристики, такі як стиль написання чи використання спеціальних символів. Це значно покращує результати класифікації навіть для неоднозначних текстів.

Процес реалізації UFTR включає попередню обробку тексту, де дані повідомлень очищуються та готуються до аналізу. Формується уніфікований набір ознак, що включає текстові характеристики, метадані та історичні дані класифікації. Модель навчається на цих даних за допомогою методів навчання з учителем, таких як градієнтний бустинг або глибокі нейронні мережі. Цей підхід дозволяє створювати алгоритми, які можуть швидко адаптуватися до нових сценаріїв класифікації текстів.

UFTR демонструє високу ефективність у задачах класифікації текстових повідомлень, де врахування контексту і метаінформації є критично важливими. Це може бути використано для систем підтримки клієнтів, де текстові запити автоматично класифікуються для перенаправлення до відповідних фахівців, або для аналізу соціальних мереж, де важливо швидко визначати категорії текстового контенту. Завдяки своїй гнучкості UFTR може адаптуватися до різноманітних форматів тексту та сценаріїв використання.

Основними перевагами UFTR у класифікації текстових повідомлень є інтеграція різних джерел даних, висока точність завдяки обліку контексту і історичних даних, а також масштабованість для обробки великих обсягів текстової інформації. Водночас, алгоритм має виклики, такі як необхідність якісних даних для навчання та оптимізація параметрів, що забезпечують високу продуктивність у складних сценаріях класифікації тексту.

1.5 Тенденції та перспективи

Класифікація текстових повідомлень є важливою складовою сучасної обробки природної мови (NLP), що має широкий спектр застосувань у бізнесі, охороні здоров'я, соціальних медіа та комунікаціях. За даними MarketsandMarkets, ринок чат-ботів оцінювався в 2,6 мільярда доларів у 2021 році та прогнозується, що він досягне 9,4 мільярда доларів до 2026 року. Це свідчить про ключову роль класифікації тексту в автоматизації обслуговування клієнтів, маркетингу та продажів. Чат-боти, побудовані на основі моделей класифікації тексту, здатні розуміти запити користувачів та надавати релевантні відповіді, що значно покращує якість обслуговування.

У соціальних медіа, де кількість користувачів у 2023 році досягла 4,7 мільярда осіб, аналіз текстових повідомлень допомагає компаніям виявляти тренди, оцінювати настрої клієнтів та підвищувати ефективність маркетингових кампаній. Використання NLP для аналізу постів, коментарів і повідомлень дозволяє швидко отримувати цінну інформацію для покращення продуктів і послуг.

Класифікація тексту також є важливою у медицині. За даними Allied Market Research, ринок NLP в охороні здоров'я зросте з 2,65 мільярда доларів у 2020 році до 16,1 мільярда доларів до 2030 року. NLP використовується для аналізу медичних записів, виявлення захворювань та вдосконалення лікування. Аналіз діалогів між лікарями та пацієнтами дозволяє ранньо діагностувати захворювання та адаптувати лікування до потреб конкретного пацієнта.

Популярність мобільних месенджерів також підкреслює актуальність класифікації текстових повідомлень. У 2023 році їх користувалися 3,51 мільярда людей, що становить 78,3% користувачів смартфонів. Аналіз тексту в месенджерах відкриває можливості для розуміння потреб користувачів, адаптації послуг та персоналізації комунікацій.

Особливу увагу в останні роки привертає класифікація тексту саме в діалогах, де важливо не лише розпізнати окреме повідомлення, а й враховувати контекст попередніх реплік. Така класифікація має критичне значення для розробки інтелектуальних діалогових систем, зокрема віртуальних асистентів, чат-ботів, голосових помічників та систем підтримки користувачів. Вона дозволяє визначити тип запиту (наприклад, інформаційний, емоційний, технічний), виявити намір користувача (інтент), класифікувати повідомлення як запитання, скаргу, подяку або спам.

У діалогах класифікація також використовується для ідентифікації зміни теми, виявлення завершення діалогу або аналізу емоційного тону розмови. Це особливо важливо для побудови систем, які можуть підтримувати природну й логічну бесіду. Наприклад, в службах підтримки клієнтів класифікація діалогів дозволяє автоматично розподіляти запити між операторами відповідної компетенції або пропонувати шаблонні відповіді ще до втручання людини.

Отже, класифікація тексту в діалогах має стратегічне значення для побудови ефективних, контекстно-орієнтованих і безпечних мовних систем, які здатні не лише розпізнавати окремі повідомлення, а й «розуміти» хід розмови загалом.

2 ОГЛЯД Й АНАЛІЗ ЛІТЕРАТУРНИХ, НАУКОВИХ ДЖЕРЕЛ

2.1 Огляд основних джерел

Огляд основних джерел для даного дослідження охоплює праці, що висвітлюють сучасні підходи до класифікації текстових повідомлень у діалогах. Однією з базових робіт стала праця Ashish Vaswani та ін. "Attention Is All You Need" [1], у якій було представлено модель трансформера. Ця модель стала революційною в обробці тексту, оскільки повністю відмовилася від рекурентних підходів і побудована на механізмі самоуваги. Її ключова перевага полягає у здатності ефективно враховувати залежності між елементами тексту, що дозволяє досягати високої точності в задачах класифікації та машинного перекладу.

Іншим важливим джерелом стала робота Adar Kahana та Oren Elisha "MESSAGENET: Message Classification Using Natural Language Processing and Meta-data" [2], у якій запропоновано багатомодальний підхід до класифікації тексту. Авторами було доведено, що інтеграція метаданих (час відправлення, автор, вкладення) із текстовими даними значно підвищує точність класифікації повідомлень. Це дослідження є особливо цінним у контексті багатофакторного аналізу текстів.

У дослідженні Jianglei Han та ін. "UFTR: A Unified Framework for Ticket Routing" [3] розроблено уніфіковану модель маршрутизації заявок. Вона враховує як текстові, так і структуровані дані, забезпечуючи комплексний підхід до вирішення задачі розподілу запитів. Особливу увагу приділено інтеграції різних джерел інформації, що дозволяє досягати високої ефективності в задачах класифікації

Ймовірнісні моделі, такі як Naïve Bayes та алгоритм Вітербі, розглянуті в роботі Justin Weisz "Segmentation and Classification of Online Chats", демонструють ефективність у задачах сегментації чатів. У цьому дослідженні аналізується, як послідовність повідомлень впливає на їх класифікацію, що є ключовим аспектом для аналізу текстових даних у реальному часі

2.2 Аналіз літератури

Дослідження сучасних підходів до класифікації текстових повідомлень у діалогах базується на вивченні основних теорій і моделей, які дозволяють аналізувати текстові дані з урахуванням контексту та метаданих. Основними концепціями, що використовуються у цій галузі, є моделі трансформерів, ймовірнісні підходи та багатомодальні архітектури, що об’єднують текстові та структуровані дані.

2.2.1 Моделі трансформерів

Фундаментом сучасного аналізу тексту є модель трансформера, представлена у роботі Ashish Vaswani та ін. "Attention Is All You Need". Ця модель ґрунтується на механізмі самоуваги, що дозволяє ефективно аналізувати залежності між елементами тексту незалежно від їхньої відстані. Відмова від рекурентних нейронних мереж забезпечила значне підвищення швидкості навчання та точності обробки текстових даних. Основними перевагами трансформерів є їх здатність враховувати глобальний контекст тексту та паралельна обробка, що робить їх ефективними для великих датасетів. Ця модель значно вплинула на розвиток задач машинного перекладу, аналізу настроїв і класифікації тексту.

2.2.2 Інтеграція метаданих

У дослідженні Adar Kahana та Oren Elisha "MESSAGENET: Message Classification Using Natural Language Processing and Meta-data" [2] розглядається підхід до класифікації текстових повідомлень, що об’єднує текстові дані з метаданими. Основна ідея полягає у створенні мультиблокової архітектури, де окремі блоки обробляють текст і метадані, а потім об’єднують ці представлення для фінальної класифікації. Наприклад, врахування інформації про час відправлення або автора повідомлення дозволяє моделі враховувати контекстуальні аспекти, які часто недоступні в тексті. Це значно підвищує

точність класифікації у задачах, де текстовий зміст недостатній для прийняття рішення.

2.2.3 Ймовірнісні підходи

У роботі Justin Weisz "Segmentation and Classification of Online Chats" були використані Naïve Bayes та алгоритм Вітербі для класифікації тексту. Ці моделі ефективно працюють із послідовними даними, такими як чати, де залежності між повідомленнями є критично важливими. Алгоритм Вітербі дозволяє знаходити оптимальну послідовність міток для текстових блоків, що особливо корисно для багатотемних чатів. Результати показують, що ймовірнісні підходи можуть бути ефективними для обробки текстів із послідовними структурами, однак поступаються трансформерам у задачах із великими обсягами даних.

2.2.4 Багатомодальні архітектури

Модель Jianglei Han та ін. "UFTR: A Unified Framework for Ticket Routing" [3] є прикладом інтеграції кількох типів даних, включаючи текстові, історичні та структуровані. Вона враховує взаємодії між заявками та групами обробки, що забезпечує точність у задачах маршрутизації. Застосування цієї моделі показало значне покращення у швидкості та точності класифікації завдяки спільному навчанню різних компонентів.

Аналогічний підхід до об’єднання різнорідних даних демонструє мультимодальна модель CLIP [4], що обробляє текстові та візуальні дані. Застосування подібного мультимодального методу дозволяє поєднувати семантику тексту з образною інформацією та розширювати можливості класифікації, пошуку і розпізнавання контенту в різноманітних сценаріях використання.

Поєднання сучасних підходів, таких як трансформери, інтеграція метаданих і багатомодальні моделі, забезпечує високий рівень продуктивності у класифікації текстових повідомлень завдяки їхній здатності враховувати як змістовні, так і контекстуальні аспекти даних. Трансформери, наприклад, дозволяють аналізувати

залежності між словами на глобальному рівні, що забезпечує точне розуміння тексту навіть у складних сценаріях. Інтеграція метаданих і багатомодальних підходів додає додаткові контекстуальні шари, такі як інформація про час або автора, що значно покращує точність класифікації у реальних застосунках.

2.3 Оцінка актуальності та новизни

Інформація, представлена в розглянутих джерелах, є актуальною та має значний вплив на сучасний стан проблеми класифікації текстових повідомлень у діалогах. Розвиток технологій обробки природної мови (NLP), зокрема впровадження трансформерів, таких як BERT, у роботі Ashish Vaswani та ін. "Attention Is All You Need" [1], дозволив суттєво підвищити продуктивність у задачах класифікації текстів завдяки здатності аналізувати довготривалі залежності між словами. Ця технологія є основою для багатьох сучасних моделей і є незамінною для завдань, де текст містить складний контекст або великі обсяги даних.

Наукова новизна роботи Adar Kahana та Oren Elisha "MESSAGENET" [2] полягає у застосуванні мультиблокової архітектури, яка об’єднує текстові дані з метаданими, такими як час, автор або вкладення. Це значно покращує точність класифікації у випадках, коли тексту недостатньо для прийняття рішень. Даний підхід відкриває нові можливості для аналізу багатофакторних даних та забезпечуючи додатковий рівень контексту.

Інтеграція структурованих даних із текстовими у моделі Jianglei Han та ін. "UFTR" [3] та медіа даних також є важливим внеском у розвиток задач маршрутизації запитів. Об’єднання інформації про взаємодії між групами обробки з текстовими описами дозволяє ефективно вирішувати завдання у багатокрокових процесах. Це підвищує ефективність не лише класифікації, але й прогнозування результатів дій системи у реальному часі.

Загалом, нові підходи, описані в розглянутих джерелах, дозволяють значно підвищити точність, ефективність і адаптивність систем класифікації текстових повідомлень. Це створює нові перспективи для застосування багатомодальних

моделей у різноманітних галузях, таких як автоматизація обробки текстів, маршрутизація запитів і класифікація повідомлень у реальному часі.

2.4 Узагальнення результатів огляду літератури

Огляд сучасних джерел демонструє, що використання трансформерів, інтеграція метаданих і багатомодальних підходів є ключовими досягненнями в класифікації текстових повідомлень. Трансформери, такі як BERT, забезпечують ефективний аналіз контексту тексту, враховуючи навіть складні залежності між словами. Інтеграція метаданих, як показано в роботах Adar Kahana та Oren Elisha і Asif Karim та ін. [2], додає додатковий рівень точності в задачах, де текстові дані недостатні для прийняття рішення. Багатомодальні архітектури, такі як у роботі Jianglei Han та ін., відкривають можливості для створення комплексних систем, що враховують текстові, структуровані й контекстуальні дані.

Ці висновки впливають на обрану тему дослідження, підтверджуючи доцільність інтеграції сучасних NLP-підходів для класифікації повідомлень у діалогах. Вони демонструють важливість багаторівневого підходу, який враховує не лише текст, але й супутні дані, такі як метадані чи історичні взаємодії.

Крім того, важливо зазначити, що використання багаторівневих моделей у класифікації діалогів дозволяє системам краще справлятися з багатозначністю реплік, зміною теми бесіди та відстеженням емоційного стану співрозмовника. Це особливо корисно в контексті автоматизованого консультування, де від точності класифікації повідомлень залежить якість підтримки користувача.

Сучасні дослідження також акцентують увагу на важливості навчання моделей на діалогових корпусах, які враховують не лише зміст окремих повідомлень, а й структурну послідовність обміну репліками. Такі підходи сприяють побудові більш природної взаємодії між системою й користувачем, що є ключовим фактором успішності інтерактивних NLP-рішень у реальному часі.

3 ПОСТАНОВКА ЗАДАЧІ

У рамках проєкту планується створити систему автоматичної класифікації текстових повідомлень, що використовує сучасні методи обробки природної мови (NLP), включаючи трансформерні моделі та мультимодальні підходи. Метою роботи є розробка ефективного інструменту для класифікації текстових даних у різних контекстах, таких як аналіз соціальних мереж, автоматизація клієнтської підтримки, медичні застосування та аналіз повідомлень у месенджерах.

Методи, які використовуються в роботі, обрані на основі їхньої актуальності та високої продуктивності, що підтверджено сучасними дослідженнями. Трансформерні моделі, зокрема такі як BERT, демонструють відмінні результати у завданнях аналізу тексту завдяки своїй здатності враховувати контекст за допомогою механізмів уваги. Також мультимодальний підхід дозволяє інтегрувати текстову інформацію з метаданими, наприклад, часовими позначками чи супутніми зображеннями, що покращує точність класифікації.

3.1 Використані технології та інструменти

Для реалізації проєкту буде використано сучасні програмні засоби. Основними інструментами стануть бібліотека HuggingFace Transformers, TensorFlow та PyTorch.

Бібліотека HuggingFace Transformers є одним із провідних інструментів для роботи з трансформерними моделями, такими як BERT, GPT-3 та RoBERTa. Вона надає доступ до попередньо навчених моделей, що дозволяє значно скоротити час на навчання та покращити якість класифікації. HuggingFace також має простий у використанні API для інтеграції моделей у різні програми, а також підтримує мультиплатформеність, що є зручною для реалізації складних проєктів.

TensorFlow є бібліотекою для роботи з глибоким навчанням, яка дозволяє створювати, навчати та оптимізувати складні моделі нейронних мереж. Вона забезпечує високий рівень паралельності завдяки підтримці обчислень на GPU та TPU. Це критично важливо для роботи з великими обсягами даних, що дозволяє пришвидшити процес навчання моделей і підвищити їхню продуктивність.

PyTorch є альтернативою TensorFlow і відома своєю простотою у використанні завдяки інтерактивності та динамічним обчислювальним графам. Ця бібліотека широко використовується у дослідницьких проєктах завдяки гнучкості в налаштуванні моделей та активній підтримці спільноти розробників. PyTorch дозволяє швидко проводити експерименти та вносити зміни до архітектури моделей.

3.2 Характеристика наборів даних

Для збору та підготовки даних буде використано кілька відкритих наборів, зокрема Multi-Domain Wizard-of-Oz (MultiWOZ) та IMDb Dataset.

MultiWOZ – це великий багатодоменний діалоговий набір даних, що містить понад 10 000 діалогів між користувачами та симульованими системами. Діалоги охоплюють кілька сфер, зокрема бронювання готелів, пошук ресторанів, замовлення таксі, інформацію про визначні місця тощо. Цей набір активно використовується для навчання моделей діалогових агентів, класифікації намірів, побудови систем ведення діалогу та автоматичного заповнення слотів.

IMDb Dataset містить понад 50 000 рецензій на фільми з оцінками (позитивна або негативна) та супровідними метаданими. У розширених версіях цього набору також доступні зображення, пов’язані з фільмами (наприклад, постери або кадри), що дозволяє використовувати його для мультимодального навчання. Такий набір дає змогу моделювати вплив візуального контексту на сприйняття тексту, що особливо корисно для завдань аналізу настроїв, тематичної класифікації чи рекомендаційних систем.

IMDb Dataset у межах цього дослідження буде використано для навчання моделі класифікації тексту в супроводі зображень, що дозволить побудувати мультимодальну систему – тобто таку, яка може враховувати як зміст тексту, так і візуальні ознаки, пов’язані з ним.

4 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

Для дослідження було застосовано три основні підходи до класифікації текстових повідомлень: трансформери, мультимодальні моделі на основі UFTR (Unified Framework for Ticket Routing) та алгоритми з використанням прихованих марковських моделей (HMM). Кожен із цих підходів характеризується унікальними особливостями, специфічними вимогами та різними сферами практичного застосування, що зумовлює їхню взаємодоповнюваність у розв’язанні задач класифікації текстових даних.

Для оцінки ефективності зазначених методів вони були порівняні з класичними алгоритмами машинного навчання, зокрема softmax-регресією, наївним баєсівським класифікатором, методом опорних векторів (SVM), методом випадкових лісів (Random Forest) та алгоритмом k-ближчих сусідів (KNN). Це порівняння дозволило не лише оцінити продуктивність обраних підходів, але й визначити їхні переваги та обмеження у порівнянні з традиційними методами класифікації.

4.1 Оцінювані параметри

Для порівняння моделей були використані такі параметри: кількість параметрів, врахування контексту (токенів), необхідний обсяг пам'яті (ГБ), розмір словника, масштабованість та інтерпретованість моделі.

4.1.1 Кількість параметрів

Трансформери, наприклад, BERT (110 млн параметрів), мають значну кількість параметрів. Це забезпечує їм високу гнучкість і здатність навчатися складним залежностям у даних. Мультимодальні моделі також мають великий обсяг параметрів. У нашому дослідженні використовується модель CLIP (ViT-B/32), яка має 112 млн параметрів.

HMM і Softmax-регресія мають менше параметрів у порівнянні з великими моделями, такими як трансформери, але кількість цих параметрів може значно зростати залежно від кількості станів, спостережень та ознак. У HMM кількість

параметрів визначається формулою $N^2 + N * M$, де N – це кількість станів у моделі, а M – кількість можливих спостережень. Формула $N^2 + N * M$ враховує матрицю переходів між станами, оскільки для кожного стану є N можливих переходів у інші N стани. Компонент $N * M$ описує емісійні ймовірності, які визначають, як кожен стан породжує спостереження.

У Softmax-регресії кількість параметрів розраховується як $N * C + C$, де N – це кількість ознак у даних, а C –кількість класів. Ця формула виникає через те, що для кожної ознаки потрібно зберігати вагу для кожного класу (тобто $N * C$), а також потрібно додати C зсувів, що є окремими параметрами для кожного класу.

Наївний Байєс, SVM, Random Forest і KNN мають відносно меншу кількість параметрів, що робить їх менш вимогливими до обчислювальних ресурсів. У Наївного Байєса кількість параметрів визначається як $N * C$, де N – кількість ознак, а C – кількість класів. Це пов’язано з тим, що алгоритм окремо обчислює ймовірність кожної ознаки для кожного класу. У SVM кількість параметрів залежить від кількості опорних векторів, які визначають межі між класами. Чим більше опорних векторів, тим більше параметрів, оскільки для кожного опорного вектора зберігається коефіцієнт впливу.

Для Random Forest кількість параметрів визначається як $T * D$, де T – це кількість дерев у лісі, а D – кількість ознак, які використовуються в кожному дереві. Кожне дерево навчається незалежно, тому параметри залежать від структури кожного дерева і їх загальної кількості. KNN взагалі не має параметрів у класичному розумінні, оскільки для класифікації цей метод зберігає всі навчальні дані. Він порівнює кожну точку з іншими під час класифікації, що означає, що основний обсяг обчислень припадає на етап передбачення.

4.1.2 Використання контексту

Трансформери здатні враховувати великий контекст, що дозволяє їм аналізувати довгі тексти та складні залежності між словами . Врахування контексту в мультимодальних моделях залежить від конкретної архітектури текстового енкодера, який використовується в моделі.

Сховані марковські моделі (HMM) [4] враховують контекст шляхом моделювання залежностей між схованими станами та спостереженнями у послідовностях даних. У HMM кожне спостереження залежить від схованого стану, а кожен стан залежить від попереднього, що описується ймовірностями переходів між станами та ймовірностями емісій. Основними параметрами HMM є кількість схованих станів (N), кількість можливих спостережень (M), матриця ймовірностей переходів між станами (A = $\{a_{ij}\}$), матриця ймовірностей емісій (B=$\{b_{ij}\}$) та початкові ймовірності станів $\pi = \{\pi_{ij}\}$. Ймовірності переходів між станами визначаються як $a_{ij} = P(z_t = j | z_{t-1} = i)$, що відображає залежність поточного стану z_t від попереднього z_{t-1}. Ймовірності емісій визначаються як $b_j(k) = P(x_t = k | z_t = j)$, що описує ймовірність спостереження x_t у стані z_t.

Повна ймовірність послідовності станів $z = \{z_1, z_2, \dots, z_T\}$ моделюється, як $P(z) = \pi_{z_1} \prod_{t=2}^{T} a_{z-1 z_t}$, що враховує початкову ймовірність π_{z_1} і ймовірності переходів між станами. Якщо задана послідовність станів z, то ймовірність спостережень обчислюється як $P(x|z) = \prod_{t=1}^{T} b_{z_t}(x_t)$. Повна ймовірність спостережень із врахуванням усіх можливих послідовностей станів визначається як $P(x) = \sum_z P(x|z)P(z)$, що інтегрує ймовірності для всіх можливих шляхів z.

Для врахування глобального контексту HMM використовує алгоритм "вперед-назад". На етапі "вперед" обчислюється ймовірність часткової послідовності спостережень до моменту часу t і стану j, як $a_t(j) = P\left(x_1, x_2, \dots, x_z, z_{z=j}\right) = \sum_{i=1}^{N} a_{t-1}(i) a_{ij} b_j(x_t)$. На етапі "назад" розраховується ймовірність часткової послідовності спостережень від моменту часу *t+1* до кінця, як $\beta_t(i) = P(x_{t+1}, x_{t+2}, \dots, x_T | z_t = i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$. Після цього ймовірність перебування у стані j у момент часу t визначається як $P(z_t = j|x) = \frac{a_t(j)\beta_t(j)}{\sum_{k=1}^{N} a_t(k)\beta_t(k)}$, що дозволяє врахувати як попередні, так і наступні спостереження.

HMM враховує контекст через залежності між станами та спостереженнями, що робить її ефективною для задач, де важливий порядок подій. Наприклад, у

задачах розпізнавання тексту або мовлення HMM враховує ймовірності переходів між станами, що відповідають граматичним чи синтаксичним правилам, і ймовірності емісій, що відображають найбільш імовірні слова для певного стану. Для задач, де потрібен ширший контекст, застосовуються розширені версії HMM, наприклад, моделі вищого порядку, які враховують залежність від кількох попередніх станів, або Input-Output HMM, де враховуються додаткові контекстні ознаки. Таким чином, HMM забезпечує інтеграцію локального та глобального контексту послідовності через свою математичну структуру та алгоритми оцінки ймовірностей

Інші підходи не враховують контекст безпосередньо, бо softmax-регресія – це лінійний класифікатор, який призначає ймовірності різним класам на основі лінійної комбінації ознак. Він розглядає кожен вхідний вектор незалежно, не беручи до уваги навколишніх даних. Наївний баєсівський класифікатор базується на теоремі Баєса і припущенні про умовну незалежність ознак. Він розглядає кожну ознаку окремо, ігноруючи їх взаємозв'язки та контекст. SVM знаходить гіперплощину, яка найкраще розділяє дані на класи. Хоча SVM може використовувати ядрові функції для моделювання нелінійних залежностей, він все ще фокусується на окремих точках даних, а не на їх послідовності чи контексті. Метод випадкових лісів – це ансамблевий метод, який об'єднує кілька дерев рішень. Кожне дерево прийняття рішень розглядає ознаки незалежно, не враховуючи контекст. KNN класифікує точку даних на основі класів її k найближчих сусідів. Хоча KNN враховує локальну структуру даних, він не враховує ширший контекст або послідовність даних.

4.1.3 Використання оперативної пам’яті

Використання оперативної пам'яті залежить від вибору моделі та її складності. Трансформери, особливо великі моделі, є дуже вимогливими до оперативної пам'яті. Наприклад модель, така як BERT, вимагає приблизно 11 ГБ. Це обумовлено великою кількістю параметрів і необхідністю їх зберігання та обробки під час навчання та прогнозування. Мультимодальні моделі, як-от CLIP

(ViT-B/32), мають менші вимоги до пам'яті – близько 4 ГБ, що зумовлено їх архітектурою та специфікою обробки текстових і візуальних даних.

HMM, на відміну від трансформерів і мультимодальних моделей, є відносно легким підходом з точки зору використання пам'яті. Для роботи HMM зазвичай достатньо менше ніж 1 ГБ оперативної пам'яті, оскільки ця модель має значно менше параметрів. Подібною є ситуація з Softmax-регресією, яка також потребує менше 1 ГБ пам'яті завдяки своїй простій архітектурі. Наївний Байєс є ще менш вимогливим підходом і зазвичай працює з кількома сотнями мегабайт пам'яті.

SVM, у свою чергу, залежить від кількості опорних векторів і складності задачі. У випадках, коли використовується велика кількість опорних векторів або дані мають високий розмір, потреби у пам'яті можуть значно зростати. Random Forest також має змінні вимоги до пам'яті, залежно від кількості дерев у моделі, їх глибини та розміру навчального набору. У великих задачах Random Forest може бути досить вимогливим до ресурсів. Метод KNN зберігає всі навчальні дані в пам'яті, що робить його залежним від розміру датасету та кількості ознак. У задачах із великими обсягами даних цей алгоритм може споживати значний обсяг оперативної пам'яті.

Проаналізовані дані зведені у таблицю 4.1, яка наочно демонструє використання оперативної пам’яті різними моделями.

Таблиця 4.1 – Кількість оперативної пам’яті

	Transformers (BERT)	Multimodal Models (CLIP)	HMM	Softmax Regression	Naive Bayes	SVM	Random Forest	KNN
Memory Usage (GB)	11	4	1	1	0.5	12	12	12

Для SVM, Random Forest та KNN, де використання пам’яті залежить від кількості опорних векторів, кількості і глибини дерев, а також розміру та кількості ознак у навчальному наборі відповідно, враховано максимальний визначений

обсяг пам’яті, збільшений на 1 для забезпечення точнішого моделювання найгіршого сценарію.

4.1.4 Аналіз розміру словника

Трансформери, такі як BERT, використовують вбудовані словники фіксованого розміру. У випадку з BERT розмір словника зазвичай становить від 30 000 до 50 000 токенів, що включають як цілі слова, так і підслова. Це забезпечує здатність моделі ефективно працювати з різноманітними текстами, включаючи рідкісні слова та спеціальну термінологію. Мультимодальні моделі, наприклад, CLIP, мають розмір словника, який залежить від конкретної реалізації. У CLIP для англійської мови використовується словник на 49 408 токенів. У HMM розмір словника визначається кількістю можливих спостережень або символів, які може генерувати модель. Він залежить від задачі і може бути як фіксованим, так і динамічним, наприклад, для задач розпізнавання мовлення словник може містити всі можливі фонеми.

У моделях Softmax-регресії, SVM, Random Forest і KNN вбудований словник не використовується. Ці підходи працюють із векторами ознак, які формуються різними методами, такими як "мішок слів" (bag-of-words) або TF-IDF. Розмірність векторів ознак залежить від кількості слів або n-грам, що використовуються для представлення тексту.

Аналіз представлено у таблиці 4.2. Оскільки у Hidden Markov Models (HMM) розмір словника визначається кількістю можливих спостережень, які модель здатна згенерувати, його обсяг враховується як максимальний визначений розмір словника, збільшений на 1. Для підходів, які не використовують словник, значення встановлюється рівним 0.

Таблиця 4.2 – Розмір словника для кожного з підходів.

	Transformers (BERT)	Multimodal Models (CLIP)	HMM	Softmax Regression	Naive Bayes	SVM	Random Forest	KNN
Vocabulary Size	50000	49408	50001	0	0	0	0	0

Вибір такого підходу до визначення розміру словника обумовлений необхідністю враховувати додаткові невідомі чи рідкісні спостереження під час розпізнавання послідовностей. Подібне масштабування дає змогу підвищити гнучкість моделі й точність результатів.

4.1.5 Масштабованість підходів

Масштабованість визначає здатність моделі ефективно обробляти зростаючі обсяги даних і складність задачі, що залежить від архітектури моделі, алгоритмів навчання, доступних обчислювальних ресурсів та характеристик даних. У трансформерів, наприклад, висока масштабованість обумовлена їхньою архітектурою. Вони підтримують паралельне обчислення через механізм самоуваги, який дозволяє обробляти всі токени послідовності одночасно, замість послідовної обробки, як у рекурентних мережах. Це дає обчислювальну складність у межах $O(n^2 * d)$, де n – довжина послідовності, d – розмір простору ознак. Також трансформери використовують ефективні алгоритми оптимізації, як-от Adam, який динамічно адаптує швидкість навчання для кожного параметра, забезпечуючи швидке зближення до оптимуму навіть на великих наборах даних. Крім того, трансформери підтримують розподілене навчання, що дозволяє використовувати ресурси кількох комп'ютерів одночасно, роблячи їх придатними для задач з великими наборами даних.

Мультимодальні моделі також демонструють високу масштабованість, особливо ті, що використовують механізми уваги. Завдяки цьому вони можуть

ефективно інтегрувати дані з різних джерел, таких як текст, зображення чи звук, і навчатися на великих наборах даних. Наприклад, архітектура моделі CLIP використовує механізм самоуваги для тексту та візуальних даних, дозволяючи масштабувати обробку за рахунок паралельних обчислень, подібно до трансформерів.

У Hidden Markov Models (HMM) масштабованість обмежена через їхню обчислювальну складність, яка зростає пропорційно до $O(T * N^2)$, де T - довжина послідовності, а N - кількість станів. Зі збільшенням кількості станів або обсягу даних кількість операцій зростає квадратично, що значно впливає на продуктивність при великих обсягах даних.

Softmax-регресія має середній рівень масштабованості, оскільки її обчислювальна складність залежить від кількості класів *(C)* та кількості ознак *(N)*. Формула обчислення ймовірності виглядає як $p(y = c|x) = \frac{\exp(w_c^T x)}{\Sigma_{c'} \exp\left(w_{c'}^T x\right)}$, де w_c – ваги для класу c. При великій кількості класів і ознак обчислювальні витрати на навчання та передбачення зростають, що обмежує масштабованість.

Наївний Байєс є одним із найбільш масштабованих підходів завдяки своїй простій архітектурі та можливості розпаралелювання обчислень. Його обчислювальна складність пропорційна $O(N * C)$, де N - кількість ознак, а C – кількість класів. Цей підхід дозволяє ефективно обробляти великі набори даних навіть на базових обчислювальних ресурсах.

SVM має обмеження в масштабованості через складність оптимізації нелінійного ядра. Обчислювальна складність залежить від кількості навчальних зразків (m) і становить $O(m^2)$ або $O(m^3)$ залежно від алгоритму. Зі збільшенням кількості навчальних даних розмір матриці, яка обробляється під час навчання, зростає квадратично або кубічно, що ускладнює обробку великих обсягів даних.

Random Forest демонструє хорошу масштабованість, оскільки кожне дерево у лісі будується незалежно, що дозволяє легко розпаралелювати обчислення. Його обчислювальна складність становить $O(T * d * \log(n))$, де T - кількість дерев, d -

кількість ознак, а n - кількість зразків. Це робить Random Forest ефективним для великих наборів даних.

KNN має низьку масштабованість через потребу в обчисленні відстаней між усіма парами точок у навчальному наборі. Обчислювальна складність для передбачення становить $O(n * d)$, де n - кількість навчальних зразків, а d - кількість ознак. Це призводить до значного зниження продуктивності при великих обсягах даних.

Для підрахунку ефективності масштабованості був використаний закон Амдала, який визначає максимальну теоретичну швидкодію програми або алгоритму в залежності від частки паралелізованих обчислень і обчислюється за формулою 4.1:

$$S = \frac{1}{(1-p)+\frac{p}{s}} \quad (4.1)$$

де p – частка задачі, яка може бути паралелізована,

s – кількість процесорів.

Обчислення планується виконувати, використовуючи систему з процесорм AMD Ryzen 5 5600H, яка має 6 процесорів.

Обрані значення частки задачі, яку можна разпаралелити для кожної моделі базуються на характеристиках їхньої архітектури та обчислювальних процесах, що впливають на можливість паралелізації обчислень. Для трансформерів (наприклад, BERT) значення p=0.95 було обрано через високу паралельність механізму самоуваги, який дозволяє обробляти всі токени тексту одночасно. Крім того, обчислення для кожного шару трансформера виконуються незалежно від інших, що дозволяє ефективно використовувати багатоядерні процесори та графічні процесори (GPU). У мультимодальних моделях, таких як CLIP, значення p=0.90 пояснюється подібними характеристиками, проте інтеграція текстових і візуальних даних додає дещо більше серійної частини, ніж у текстових трансформерах.

Для Hidden Markov Models (HMM) було обрано p=0.50, оскільки обчислення ймовірностей переходів між станами та емісій відбувається послідовно, що обмежує паралелізацію. Водночас, частина обчислень, наприклад, оцінка ймовірностей емісій для кожного стану, може виконуватися паралельно, що визначає половину завдання як паралелізовану. Softmax-регресія має p=0.70, оскільки обчислення для множення матриць і ймовірностей класів можуть виконуватися паралельно, але серійна частина, пов'язана з навчанням ваг, залишається обмеженням.

Наївний Байєс демонструє високу паралельність (p=0.85), оскільки ймовірності для кожної ознаки та класу обчислюються незалежно. Це дозволяє легко розпаралелювати обчислення навіть на великих наборах даних. Для SVM значення p=0.60 було обрано через серійну природу оптимізації функції ядра та визначення опорних векторів. Однак, обчислення відстаней між зразками під час передбачення може бути паралелізованим, що визначає часткову паралельність моделі. У Random Forest p=0.80, оскільки кожне дерево в лісі будується незалежно, що забезпечує високу паралельність, але об'єднання результатів між деревами додає серійної складової.

Для KNN було обрано найнижче значення p=0.30, оскільки основна частина обчислень пов'язана з послідовним обчисленням відстаней між точкою передбачення та всіма зразками навчального набору.. Підрахований рівень теоретичного прискорення підходів за рахунок масштабованості представлений у таблиці 4.3

Таблиця 4.3 – Рівень масштабованості досліджуваних підходів

	Transformers (BERT)	Multimodal Models (CLIP)	HMM	Softmax Regression	Naive Bayes	SVM	Random Forest	KNN
Розпаралелена частина (p)	0.95	0.90	0.50	0.70	0.85	0.60	0.80	0.30
Теоретичне прискорення	6.35	5.33	1.60	2.96	4.44	2.00	3.57	1.28

Отже, трансформери та мультимодальні моделі демонструють високу масштабованість завдяки своїй архітектурі та алгоритмам навчання, тоді як HMM, SVM і KNN мають значні обмеження через їхню обчислювальну складність. Random Forest і Наївний Байєс є ефективними підходами для обробки великих даних завдяки можливостям паралельної обробки та низьким витратам на обчислення.

4.1.6 Рівень інтерпретованості моделей

Інтерпретованість визначає, наскільки легко зрозуміти процес прийняття рішень моделлю, що дозволяє аналізувати вплив різних факторів на результат і виявляти потенційні проблеми, такі як упередження або помилки в даних. Трансформери, хоча й забезпечують високу точність, мають низьку інтерпретованість через складну архітектуру та велику кількість параметрів. Аналіз впливу окремих параметрів і механізмів уваги на результати є складним завданням. Мультимодальні моделі також характеризуються низькою інтерпретованістю, оскільки вони обробляють дані з різних джерел і використовують складні механізми для інтеграції цих даних.

На відміну від цього, HMM мають середній рівень інтерпретованості. Вони дозволяють аналізувати ймовірності переходів між станами та емісій для кожного стану, що допомагає зрозуміти, як модель генерує послідовності. Softmax-регресія демонструє високу інтерпретованість, оскільки ваги, пов’язані з кожною ознакою, відображають її вплив на ймовірність приналежності до кожного класу. Аналогічно, Наївний Байєс також має високу інтерпретованість завдяки можливості аналізу ймовірностей ознак для кожного класу, що дозволяє ідентифікувати найважливіші ознаки для класифікації.

SVM характеризується середнім рівнем інтерпретованості. У цій моделі можна аналізувати опорні вектори, які визначають межі рішень, і їх ваги, що допомагає зрозуміти, які точки даних є ключовими для класифікації. Random Forest має низьку інтерпретованість, оскільки використовує ансамбль дерев рішень, де вплив окремих дерев на остаточний результат є важким для аналізу.

KNN, навпаки, має високу інтерпретованість, оскільки дозволяє проаналізувати k-найближчих сусідів до точки даних та їх класи, що дає змогу зрозуміти логіку прийняття рішень моделлю. Таким чином, вибір моделі залежить від балансу між точністю, інтерпретованістю та іншими вимогами до задачі.

Рівень інтерпретованості для кожного підходу було оцінено за шкалою: низький – 1, середній – 2, високий – 3, результати наведені в таблиці 4.4.

Таблиця 4.4 – рівні інтерпритованості моделей

	Transformers (BERT)	Multimodal Models (CLIP)	HMM	Softmax Regression	Naive Bayes	SVM	Random Forest	KNN
Memory Usage (GB)	2	2	1	2	3	1	2	3

Отже, трансформери та мультимодальні моделі забезпечують високу точність та добре масштабуються, але їх інтерпретованість низька. HMM може бути ефективним для аналізу послідовностей, але має обмеження у врахуванні контексту та масштабованості. Softmax-регресія та Наївний Байєс є простими та ефективними підходами з високою інтерпретованістю, але можуть уступати в точності. SVM та Random Forest є потужними алгоритмами, але можуть потребувати більше пам'яті. KNN є простим та швидким алгоритмом з високою інтерпретованістю, але його ефективність може знижуватися при великій кількості ознак та обмежена в масштабованості.

4.2 Проведення теоретичного дослідження

Для теоретичного дослідження розглянемо підходи та параметри, описані у попередньому розділі: UFTR (Unified Framework for Ticket Routing) як мультимодальну модель, HMM, трансформер BERT, Softmax-регресія, наївний баєсівський класифікатор, метод опорних векторів (SVM), метод випадкових лісів (Random Forest) та алгоритм k-ближчих сусідів (KNN). Ці моделі будуть

порівнюватися за такими показниками, як кількість параметрів, врахування контексту (токенів), необхідний обсяг пам'яті (ГБ), розмір словника, масштабованість та інтерпретованість.

Складемо векторний опис обраних для дослідження моделей за визначеними критеріями, результат наведений у таблиці 4.5.

Таблиця 4.5 – Векторне представлення алгоритмів класифікації тексту

	Кількість параметрів	Врахування контексту (токенів)	Необхідна пам'ять (ГБ)	Розмір словника	Масштабованість	Інтерпретованість
Transformers (BERT)	110 млн	512	11	50000	6.35	2
Multimodal Models (CLIP)	112 млн	512	4	49408	5.33	2
HMM	112 млн	512	1	50001	1.60	1
Softmax Regression	112 млн	0	1	0	2.96	2
Naive Bayes	112 млн	0	0.5	0	4.44	3
SVM	112 млн	0	12	0	2.00	1
Random Forest	112 млн	0	12	0	3.57	2
KNN	0	0	12	0	1.28	3

Показник необхідної пам’яті відображає обсяг ресурсів, потрібних для роботи алгоритму. Оскільки менше споживання пам’яті свідчить про кращу ресурсоємність підходу, цей параметр буде перетворений до показника оптимальності за принципом: чим менший обсяг пам’яті використовується, тим вищий рівень її економічності. Перетворений результат наведений в таблиці 4.6.

Порівняємо множину отриманих альтернатив. HMM залишається в аналізі завдяки своїй здатності працювати з послідовностями та враховувати контекст, що є критичним для задачі. Хоча масштабованість HMM є нижчою порівняно з

іншими підходами, її ефективність у задачах, пов’язаних із ймовірнісними переходами між станами, компенсує цей недолік.

Таблиця 4.6 – Приведення показників по принципу оптимальності «за максимумом»

	Кількість параметрів	Врахування контексту (токенів)	Збережена пам’ять (ГБ)	Розмір словника	Масштабованість	Інтерпретованість
Transformers (BERT)	110 млн	512	1	50000	6.35	2
Multimodal Models (CLIP)	112 млн	512	8	49408	5.33	2
HMM	112 млн	512	11	50001	1.60	1
Softmax Regression	112 млн	0	11	0	2.96	2
Naive Bayes	112 млн	0	11.5	0	4.44	3
SVM	112 млн	0	0	0	2.00	1
Random Forest	112 млн	0	0	0	3.57	2
KNN	0	0	0	0	1.28	3

KNN демонструє найнижчий рівень масштабованості і не враховує контекст (0 токенів). Це робить його непридатним для задач, які потребують обробки послідовних або контекстуальних даних. Крім того, велика обчислювальна складність під час обробки великих обсягів даних через необхідність обчислення відстаней між усіма точками значно обмежує його застосування.

Random Forest, хоча і має середню масштабованість та можливість паралельної обробки, не враховує контекст (0 токенів) і залежить від великої кількості дерев, що збільшує споживання пам’яті та може призводити до нестабільності результатів на складних даних. У задачах, де врахування контексту є критичним, цей підхід не є оптимальним.

SVM, із масштабованіст, також поступається більш сучасним підходам, оскільки не враховує контекст і має залежність від кількості опорних векторів, що ускладнює його використання для великих наборів даних. Хоча SVM забезпечує потужну класифікацію на невеликих наборах даних, у масштабних і контекстуально залежних задачах він менш ефективний.

Таким чином, за принципом Парето, KNN, Random Forest і SVM доцільно виключити з подальшого аналізу через їхню нездатність працювати з контекстом і обмежену масштабованість, що є важливими критеріями для задач класифікації текстових повідомлень. Трансформери, мультимодальні моделі, HMM, Softmax-регресія та Наївний Байєс залишаються оптимальними кандидатами для розв'язання поставленої задачі.

Проведемо нормування оцінок за шкалами. Формула 4.2 для нормування:

$$f = \frac{f_{\text{вимір}} - f_{min}}{f_{max} - f_{min}} \qquad (4.2)$$

де f_{min} – мінімальна з оцінок критерію,

f_{max} – максимальна з оцінок критерію,

f_{max} – оцінка для якої відбувається нормалізація.

Результати після нормування наведені у таблиці 4.7.

Таблиця 4.7 – Нормування значень за параметрами

	Кількість параметрів (млн)	Врахування контексту (токенів)	Збережена пам’ять (ГБ)	Розмір словника	Масштабованість	Інтерпретованість
Transformers (BERT)	0	1	0	0.99	1	0.5
Multimodal Models (CLIP)	1	1	0.68	0.98	0.78	0.5

Кінець таблиці 4.7

	Кількість параметрів (млн)	Врахування контексту (токенів)	Збережена пам’ять (ГБ)	Розмір словника	Масштабованість	Інтерпретованість
HMM	1	1	0.7	1	0	0
Softmax Regression	1	0	0.7	0	0.28	0.5
Naive Bayes	1	0	1	0	0.59	1

У таблиці подано нормалізовані оцінки різних моделей за такими шістьма ключовими параметрами, як кількість параметрів, врахування контексту, збережена пам’ять, розмір словника, масштабованість та інтерпретованість.

Для визначення моделей, які найбільш відповідають заданим критеріям, було використано метод лінійної адитивної згортки з ваговими коефіцієнтами, оскільки деякі критерії суттєво впливають на ефективність та вибір моделі (формула 4.3).

$$Z^* = \max_{i=1,m} \sum_{j=1}^{n} a_j b_j a_{ij} \tag{4.3}$$

де a_j – нормуючий множник,

β_j – ваговий коефіцієнт.

Для визначення нормуючого множника використовується формула 4.4.

$$a_i = \frac{1}{\sum_{i=1}^{m} a_{ij}} \tag{4.4}$$

Кількість параметрів, що визначає потужність моделі, отримала вагу 3. Хоча більша кількість параметрів зазвичай свідчить про вищу здатність моделі до навчання, вона також підвищує ресурсоємність і час навчання, що робить цей параметр середньо важливим. Врахування контексту, який є ключовим фактором для розуміння тексту, отримало найвищу вагу – 5, оскільки саме цей параметр

визначає, наскільки добре модель здатна працювати з семантичними зв’язками та складними структурами тексту.

Збережена пам’ять отримала вагу 4, адже ефективне використання пам’яті є критично важливим у реальних умовах, де ресурси можуть бути обмеженими. Розмір словника, що визначає здатність моделі працювати з різноманітними текстами, отримав вагу 3. Великий словник може бути корисним для аналізу текстів зі специфічною термінологією, але він також підвищує ресурсоємність, тому його важливість оцінена як середня.

Масштабованість, яка є ключовим фактором для роботи з великими наборами даних або складними задачами, також отримала найвищу вагу – 5. Цей параметр є критично важливим для забезпечення ефективності моделі у багатопроцесорному середовищі. Інтерпретованість, хоча й важлива для пояснення роботи моделі, отримала найнижчу вагу – 2, оскільки у задачах, де точність і масштабованість мають вищий пріоритет, цей параметр не є вирішальним. У сумі отримали 22. Результати підрахунку згортки представлені у таблиці 4.8.

Таблиця 4.8 – Результати розрахунку згортки досліджуваних моделей

	Кількість параметрів (млн)	Врахування контексту (токенів)	Збережена пам’ять (ГБ)	Розмір словника	Масштаб.	Інтерпр.	Згортка
Transformers (BERT)	0	1	0	0.99	1	0.5	0.94533
Multimodal Models (CLIP)	1	1	0.68	0.98	0.78	0.5	0.8466
HMM	1	1	0.7	1	0	0	0.5980
Softmax Regression	1	0	0.7	0	0.28	0.5	0.4786
Naive Bayes	1	0	1	0	0.59	1	0.4173

Результати згортки моделей показують чіткий поділ на три групи за їх ефективністю. Найвищі результати продемонстрували трансформери (BERT) із згорткою 0.94533 та мультимодальні моделі (CLIP) із значенням 0.8466. Їхні високі показники пояснюються здатністю ефективно обробляти контекст (512 токенів), значною масштабованістю та великою кількістю параметрів (110–112 млн). Ці підходи є найкращими для задач, що потребують високої точності та обробки великих обсягів даних.

HMM із результатом 0.5980 займає середню позицію, демонструючи збалансовані характеристики. Модель добре працює з послідовностями, враховує контекст (512 токенів), але поступається за масштабованістю та потребує значного обсягу пам’яті (11 ГБ). Попри ці обмеження, HMM залишається актуальним для задач із ймовірнісною структурою даних. У той же час Softmax Regression із результатом 0.4786 та Naive Bayes із 0.4173 опинилися в нижній групі. Це зумовлено їхніми обмеженими можливостями врахування контексту (0 токенів) та середньо-низькою масштабованістю (2.96 для Softmax Regression і 4.44 для Naive Bayes). Хоча ці моделі є простими у реалізації та мають високу інтерпретованість, їх ефективність значно поступається сучасним методам.

Таким чином, для подальших досліджень доцільно зосередитися на моделях із високими та середніми значеннями згортки: трансформерах (BERT), мультимодальних моделях (CLIP) та HMM. Вони забезпечують оптимальний баланс між точністю, врахуванням контексту та масштабованістю. Моделі з низькими результатами, такі як Softmax Regression та Naive Bayes, можуть бути виключені з подальшого аналізу через їхні обмежені можливості для складних задач класифікації тексту.

Отже для подальшого дослідження будуть використовуватися підходи із першої та другої групи, а саме трансформер BERT, мультимодальна модель CLIP та HMM.

5 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

5.1 Огляд використаних наборів даних

У рамках практичного дослідження було обрано два різнорідні набори даних: MultiWOZ (багатодоменний діалоговий корпус) та IMDb Dataset (текстово-візуальний корпус рецензій на фільми). Обидва набори використовуються для тренування і тестування моделей машинного навчання.

5.1.1 MultiWOZ

MultiWOZ (Multi-Domain Wizard-of-Oz) – це один із найбільших відкритих багатодоменних корпусів діалогів, який охоплює понад 10 000 діалогів між користувачами та симульованими системами. Набір створений для досліджень у сфері систем ведення діалогів, де важливо не тільки коректно розпізнати намір користувача, а й зберегти контекст розмови для коректної відповіді.

Кожен діалог складається з послідовних реплік користувача й відповіді системи. В анотації міститься інформація про інтенції, заповнені слоти (наприклад, тип закладу, адреса, час, дата) та активні сервіси, з якими пов’язана конкретна репліка. Тематика охоплює бронювання готелів, пошук ресторанів, замовлення таксі, отримання інформації про події тощо. Для кожного діалогу передбачено детальну розмітку, що дає змогу виконувати класифікацію інтенцій користувача, виокремлювати й заповнювати потрібні слоти, а також оцінювати «стан діалогу», який має відстежувати система.

Завдяки великому обсягу й детальній структурі корпус MultiWOZ дає можливість перевіряти ефективність моделей, що одночасно працюють із кількома доменами, тестувати алгоритми на здатність зберігати глобальний контекст діалогу та порівнювати різні архітектури (рекурентні мережі, трансформери, HMM тощо) за точністю й здатністю до узагальнення. Завдяки такому підходу можна виявляти сильні й слабкі сторони моделей, які орієнтовані на послідовну обробку реплік та врахування контексту розмови, що робить MultiWOZ стандартом де-факто для розробників діалогових систем.

5.1.2 IMDb Dataset

IMDb Dataset – це великий набір текстових рецензій на фільми, який у розширених версіях містить також пов’язані зображення (афіші, кадри з фільмів тощо). Він широко використовується для завдань аналізу настроїв (Sentiment Analysis), категоризації тексту, а також для експериментів із мультимодальними підходами, де текстова інформація доповнюється візуальними ознаками.

Базова версія датасету включає понад 50 000 текстових рецензій, кожна з яких позначена відповідною оцінкою (позитивна чи негативна). У мультимодальних розширеннях містяться зображення, пов’язані з контентом фільму, що дозволяє проводити спільний аналіз тексту і візуальних характеристик. Оскільки рецензії часто містять специфічний сленг, різноманітні теми та неоднорідні за довжиною тексти, IMDb Dataset є зручним набором даних для тестування стійкості та узагальнювальної здатності моделей. Для мультимодальних підходів, де враховуються візуальні дані, цей датасет унікальний завдяки можливості співставити конкретні емоційні характеристики тексту з образотворчими елементами.

Попри те, що IMDb Dataset не містить діалогів у прямому сенсі, він добре підходить для тренування або донавчання моделей, які згодом можна адаптувати до класифікації діалогів. По-перше, великий обсяг різностильових текстів та багате лексичне покриття сприяють формуванню загальної мовної моделі. По-друге, у варіанті з мультимедійними матеріалами датасет відкриває можливості мультимодального навчання, що є корисним у випадках, коли в діалогах присутні додаткові зображення.

5.1.3 Обробка та підготовка даних

Ефективність моделей машинного навчання великою мірою залежить від якості та повноти вхідних даних, а також від правильної процедури їх підготовки. У цьому проєкті для роботи з двома основними наборами (MultiWOZ та IMDb) було передбачено низку уніфікованих етапів обробки, таких як первинне очищення від дублікатів та зайвих символів, токенізація та нормалізація тексту, а

також розбиття на тренувальну, валідаційну та тестову вибірки. Водночас кожен датасет має власну специфіку: у MultiWOZ слід було зберегти послідовність реплік діалогу, аби не порушити контекст, а в IMDb інколи доводилося скорочувати надмірно великі рецензії, щоб дотриматися ліміту токенів.

Для роботи з моделлю BERT було створені функції, що готують діалоги у вигляді тензорів з урахуванням максимальної кількості токенів та максимальної кількості реплік. Відповідний код представлений у додатку А.

Для HMM та Наївного Байєса підхід до структурування діалогів полягає в послідовному аналізі реплік та уточненні сервісу, який активний на кожному кроці (див. дод. Б та В). Використовувався код, де основні дії зводяться до завантаження всіх JSON-файлів, вилучення валідних послуг і обробки мовлення у кожній репліці.

Для мультимодального навчання застосовувалися функції, що дають змогу обробляти JSON-файли з інформацією про жанри фільмів та формувати унікальний набір жанрів. Після цього створюється відображення genre2idx, яке можна використовувати для навчання моделі, що враховує водночас текстові й візуальні дані:

```
folder = "dataset"
json_files = [f for f in os.listdir(folder) if f.endswith('.json')]
unique_genres = set()
for file in json_files:
    file_path = os.path.join(folder, file)
    with open(file_path, 'r') as f:
        info = json.load(f)
    genres = info.get("genres", [])
    for genre in genres:
        unique_genres.add(genre)
all_genres = sorted(list(unique_genres))
genre2idx = {genre: idx for idx, genre in enumerate(all_genres)}
idx2genre = {idx: genre for genre, idx in genre2idx.items()}
```

Таким чином, кожен із перелічених етапів та скриптів допомагає узгоджено обробити як текстові, так і мультимодальні дані

5.2 Моделі для класифікації в діалогах (на основі MultiWOZ)

MultiWOZ містить реальні багатодоменні діалоги з детальною анотацією, саме цей датасет було обрано як базову платформу для розгортання та тестування різних моделей класифікації. Головна мета полягала в тому, аби оцінити здатність алгоритмів не лише розпізнавати окремі наміри користувачів, а й враховувати контекст, що є критичним у розмовних системах.

Одним із підходів, які застосовувалися для обробки діалогів, є модель на основі трансформера BERT. Завдяки механізму самоуваги BERT здатний ефективно виявляти тривалі послідовності реплік, підтримуючи двонаправлений аналіз тексту.

Іншим підходом виступили Приховані Марковські моделі (HMM), що дають змогу враховувати послідовність станів і спостережень у діалозі. У MultiWOZ кожна репліка може належати певному сервісу або категорії. HMM підходить до цього формату, оскільки можуть моделювати імовірнісні переходи від однієї репліки до іншої, намагаючись виявити послідовні патерни. Такі моделі демонструють позитивні результати в тих випадках, коли існує ланцюгова природа діалогу, проте можуть поступатися трансформерам у складніших сценаріях, де контекст розтягується на кілька десятків реплік.

Для врахування додаткових медіданих використовувалися мультимодальні підходи. Для цього застосовувалася архітектура CLIP аби поєднати текст із супутніми ознаками, наприклад фотографіями.

Паралельно з сучасними архітектурами (BERT, HMM та мультимодальними трансформерами) було проведено експерименти з простішими підходами**,** зокрема Наївним Байєсом у поєднанні з TF-IDF та багатомітковою (One-vs-Rest) класифікацією. Результати показали, що наївний Байєс може досить швидко навчатися та давати прийнятну точність у тих випадках, коли діалоги не перенасичені складним контекстом

5.2.1 BERT-модель

BERT (Bidirectional Encoder Representations from Transformers) застосовує двонаправлену трансформер-архітектуру, що дає моделі змогу одночасно аналізувати контекст слів зліва направо й справа наліво. Спершу текст розбивається на токени (зокрема, із використанням підсловної токенізації на кшталт WordPiece), до них додають спеціальні токени на зразок [CLS] і [SEP], а також позиційні та сегментні ембеддинги. Далі механізм самоуваги (self-attention) у кожному шарі обчислює коефіцієнти уваги для кожного токена відносно інших, що дозволяє виявляти складні залежності у всій послідовності. Під час попереднього навчання BERT використовує метод маскованого мовного моделювання (Masked Language Model), маскуючи деякі слова та намагаючись їх відтворити, а також перевіряє логічний зв’язок між двома реченнями (Next Sentence Prediction). У підсумку модель генерує векторні представлення для кожного токена, а особливе місце займає токен [CLS], з якого потім можна зчитувати узагальнену інформацію для класифікації чи інших завдань. Завдяки двонаправленому контексту та можливості тонкого налаштування (fine-tuning) BERT демонструє високу точність у широкому спектрі задач обробки природної мови, легко адаптуючись до конкретних датасетів або сценаріїв використання. На рисунку 5.1 зображена загальна архітектура моделі BERT.

На зображені представлена схему трансформер-архітектури, що складається з двох головних частин: енкодера і декодера. Для завдань типу тих, що розв’язує BERT, використовується переважно енкодер: BERT складається з кількох послідовних енкодерних шарів, тоді як декодерна частина, яка відповідає за генерацію вихідної послідовності, в цій моделі не задіяна. В енкодері кожен шар має дві ключові компоненти: механізм самоуваги (Multi-Head Self-Attention) та багатошаровий перцептрон (Feed-Forward Network). Перший компонент обчислює, наскільки кожне слово (токен) у реченні «уважний» до решти токенів, завдяки чому модель враховує всі взаємозв’язки між словами незалежно від їхнього порядку.

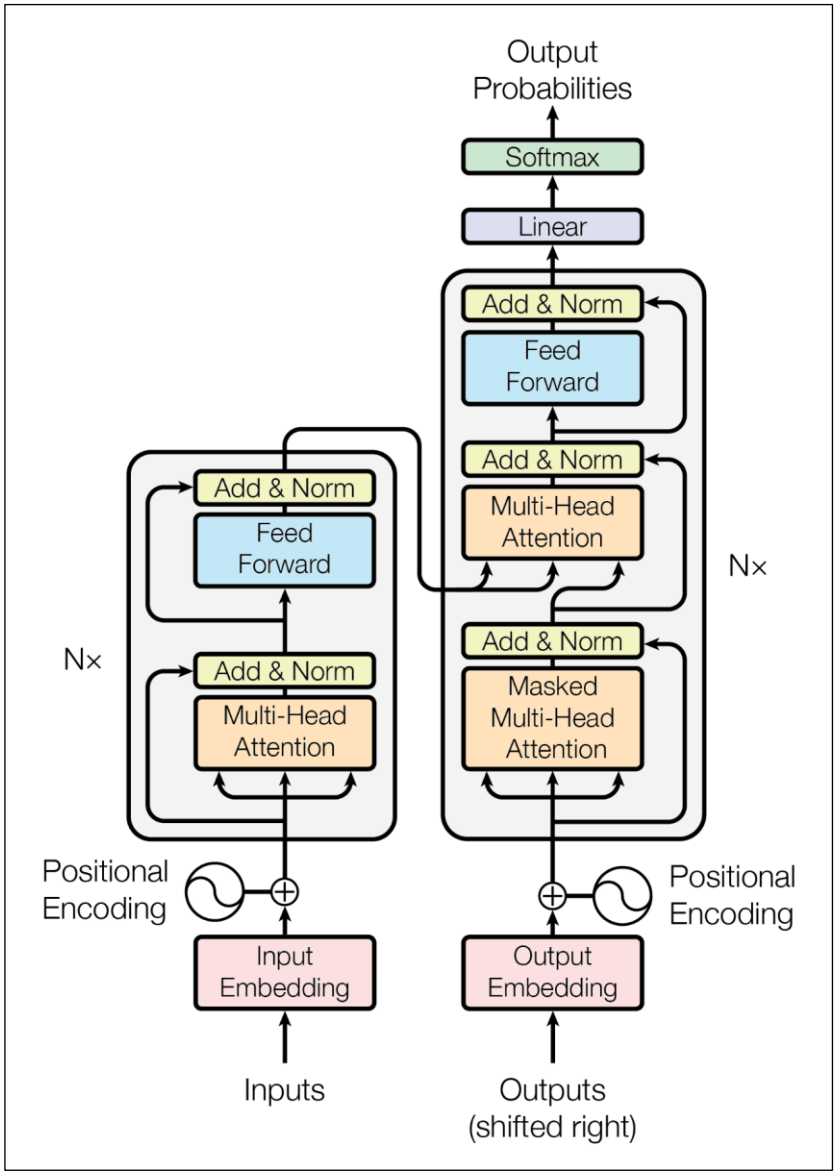

Рисунок 5.1 – Загальна архітектура BERT (https://quantpedia.com/bert-model-bidirectional-encoder-representations-from-transformers/)

Другий компонент (Feed-Forward Network) відповідає за подальшу нелінійну трансформацію та узагальнення обчисленої інформації. У підсумку багаторівневий енкодер поетапно трансформує вихідні токени у високорівневі семантичні вектори, з яких BERT може зчитувати різноманітні мовні патерни. Важливим є те, що всі ці обчислення відбуваються одночасно (паралельно) для всієї послідовності, що дозволяє BERT ефективно обробляти навіть довгі речення, а завдяки маскованому навчанню (Masked Language Model) модель засвоює залежності як зліва направо, так і справа наліво в межах одного шару.

5.2.1.1 BERT для багатозначної класифікації

Оскільки один діалог у наборі даних MultiWOZ може одночасно належати до кількох категорій, була розроблена модель, здатна класифікувати послідовність реплік за кількома типами одночасно. У наведеному коді представлено підхід, де одна й та сама модель здатна паралельно класифікувати як репліки (turns) у діалозі, так і визначати загальні категорії на рівні всього діалогу. Для цього був створений клас MultiTaskHierarchicalBERTClassifier, код якого представлений нижче.

```
    class MultiTaskHierarchicalBERTClassifier(nn.Module):
        def __init__(self, bert_model_name: str, rnn_hidden_size: int,
num_labels: int):
            super(MultiTaskHierarchicalBERTClassifier, self).__init__()
            self.bert = BertModel.from_pretrained(bert_model_name)
            self.utterance_classifier                                    =
nn.Linear(self.bert.config.hidden_size, num_labels)
            self.gru = nn.GRU(
                input_size=self.bert.config.hidden_size,
                hidden_size=rnn_hidden_size,
                num_layers=1,
                batch_first=True,
                bidirectional=True
            )
            self.dialogue_classifier  =  nn.Linear(rnn_hidden_size  *  2,
num_labels)
```

У цьому класі реалізовано модель, яка поєднує BERT-енкодер, класифікатори та рекурентну нейронну мережу для обробки діалогів у багатомітковому форматі.

Першим компонентом є загальний BERT-енкодер (`self.bert`), завантажений на основі попередньо навченого чекпойнту (наприклад, `bert-base-uncased`). Для кожної репліки в діалозі BERT генерує векторне представлення. Особливу роль відіграє [CLS]-токен, який вважається узагальненим вектором для всього вхідного тексту репліки.

Далі, класифікатор на рівні реплік (`self.utterance_classifier`) приймає ці [CLS]-ембеддинги як вхід і видає логіти – числові значення, що

відповідають ймовірності належності до кількох сервісів одночасно. Це дозволяє здійснювати багатоміткове передбачення для кожної окремої репліки.

Для урахування зв’язків між репліками використовується GRU-модуль (`self.gru`) – двонапрямлена рекурентна нейронна мережа, яка агрегує ембеддинги в послідовності. Такий підхід дає змогу враховувати як попередній, так і наступний контекст у межах діалогу, що є критично важливим для розуміння загального сенсу розмови.

Нарешті, класифікатор на рівні діалогу (`self.dialogue_classifier`) працює з фінальним станом GRU та виконує багатоміткову класифікацію вже не окремих реплік, а повного діалогу. Це дає змогу системі робити узагальнені висновки про весь діалог, спираючись на контекст і внутрішні зв’язки між репліками.

Додатково в цьому класі реалізований метод forward для обчислення ембеддингів для кожної репліки, а також формування ланцюга GRU з цих ембеддингів. Його код представлений нижче.

```
    def forward(self, input_ids, attention_mask, dialogue_labels=None, utterance_labels=None):
        batch_size, num_turns, seq_len = input_ids.size()
        flat_input_ids = input_ids.view(-1, seq_len)
        flat_attention_mask = attention_mask.view(-1, seq_len)
        outputs = self.bert(input_ids=flat_input_ids, attention_mask=flat_attention_mask)
        cls_embeddings = outputs.last_hidden_state[:, 0, :]
        turn_embeddings = cls_embeddings.view(batch_size, num_turns, -1)

        utterance_logits = self.utterance_classifier(turn_embeddings)  # (batch_size, num_turns, num_labels)

        gru_output, hidden = self.gru(turn_embeddings)
        hidden_cat = torch.cat((hidden[-2], hidden[-1]), dim=1)    # (batch_size, rnn_hidden_size*2)
        dialogue_logits = self.dialogue_classifier(hidden_cat)     # (batch_size, num_labels)
        output = {"dialogue_logits": dialogue_logits, "utterance_logits": utterance_logits}
        loss = None
        loss_fct = nn.BCEWithLogitsLoss()
        if dialogue_labels is not None and utterance_labels is not None:
            dialogue_loss = loss_fct(dialogue_logits, dialogue_labels.float())
            utterance_loss = loss_fct(utterance_logits, utterance_labels.float())
```

```
            loss = dialogue_loss + utterance_loss
            output["loss"] = loss
        return output
```

Спочатку BERT обробляє кожен реплыку (`turn`) діалогу у пласкому вигляді (батч з `num_turns` перетворюється на розмірність (`batch_size * num_turns, seq_len`)). Потім, отримані [CLS]-вектори відновлюються у форму (`batch_size, num_turns, hidden_size`). Саме до них застосовується `self.utterance_classifier`, що дає `utterance_logits`, а також вони передаються в GRU для агрегування (діалогового прогнозу).

Для обчислення втрат модель використовує `nn.BCEWithLogitsLoss()` і сумує `dialogue_loss` з `utterance_loss`, якщо на вхід подані мітки. Таким чином, BERT одночасно навчається коректно класифікувати репліки та враховувати загальну категорію діалогу.

Для підготовки даних використовуються функції, які наведені у додатку А. Варто звернути увагу на функцію `prepare_dialogue`, яка використовує `BertTokenizer` для токенізації кожної репліки до `max_seq_length` символів (з урахуванням паддінгу й усічення), при цьому загальна кількість реплік не перевищує `max_turns`. Якщо у діалозі реплік менше, вони доповнюються порожніми.

Для зручного використання фреймворку PyTorch визначено клас `MultiTaskDialogueDataset`, який визначає набір даних.

```
class MultiTaskDialogueDataset(Dataset):
    def __init__(self, dialogues, dialogue_labels, utterance_labels,
tokenizer, max_turns: int = 10, max_seq_length: int = 64):
        self.dialogues = dialogues
        self.dialogue_labels = dialogue_labels
        self.utterance_labels = utterance_labels
        self.tokenizer = tokenizer
        self.max_turns = max_turns
        self.max_seq_length = max_seq_length

    def __len__(self):
        return len(self.dialogues)

    def __getitem__(self, idx):
        dialogue_turns = self.dialogues[idx]
        d_label = self.dialogue_labels[idx]
```

```
        u_labels = self.utterance_labels[idx]
        input_ids, attention_mask = prepare_dialogue(
            dialogue_turns,         self.tokenizer,         self.max_turns,
self.max_seq_length
        )
        return {
            "input_ids": input_ids,
            "attention_mask": attention_mask,
            "dialogue_labels": torch.tensor(d_label, dtype=torch.float),
            "utterance_labels": torch.tensor(u_labels, dtype=torch.float)
        }
```

У методі `__getitem__` повертаються тензори `input_ids`, `attention_mask`, `dialogue_labels` та `utterance_labels`, готові до подачі в модель. Для тренування викликається код:

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
)
trainer.train()
```

Завдяки ньому модель послідовно оптимізує ваги BERT, лінійних шарів класифікації та GRU, узгоджуючи їх між собою.

Таким чином, наведена реалізація демонструє спосіб використання BERT у багатозадачному форматі, коли модель одночасно вчиться робити висновки про репліки (які сервіси вони представляють) та про цілий діалог (які сервіси були активними загалом). За рахунок додавання GRU на рівні діалогу BERT розширює свої можливості: він не лише бере до уваги двонаправлений контекст на рівні кожної репліки, а й акумулює послідовність усіх реплік, що корисно при зміні теми діалогу.

5.2.1.2 BERT для однозначної класифікації

Під час однозначної класифікації кожна репліка матиме лише один правильний сервіс. Такий підхід спрощує і пришвидшує навчання, але робить модель менш точною в порівняні з багатозначною класифікацією. Для підготовки даних ми використовуємо спрощену версію коду з додатку А, а саме для кожного

діалогу беремо перший сервіс як мітку для всіх реплік користувача. Після цього ми створюємо відображення імен сервісів у числові індекси.

```
texts = []
labels = []
services_set = set()

for dialogue in dialogues:
    if dialogue.get("services"):
        primary_service = dialogue["services"][0]
        services_set.add(primary_service)
        for turn in dialogue.get("turns", []):
            if turn.get("speaker") == "USER":
                texts.append(turn.get("utterance", ""))
                labels.append(primary_service)

service2label = {service: idx for idx, service in enumerate(sorted(services_set))}
numeric_labels = [service2label[label] for label in labels]
```

Тут масив `texts` містить текстові репліки користувача (USER), а `labels` – відповідні сервіси. Паралельно ми збираємо всі унікальні сервіси в множині `services_set`, щоб згодом закодувати їх у вигляді `service2label`. Для токенізації та передачі даних у модель створено клас, який представлено нижче.

```
class DialogueDataset(Dataset):

def __init__(self, texts, labels, tokenizer, max_len=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            truncation=True,
            padding="max_length",
            return_attention_mask=True,
            return_tensors="pt"
        )


        return {
```

```
            "input_ids": encoding["input_ids"].squeeze(),
            "attention_mask": encoding["attention_mask"].squeeze(),
            "labels": torch.tensor(self.labels[idx]
, dtype=torch.long)}
```

Цей датасет на вхід отримує список текстів, список меток і екземпляр токенізатора. На кожному кроці він виконує токенізацію, обмежує довжину послідовності і повертає тензори `input_ids`, `attention_mask` та ціле число, яке позначає індекс сервісу. Після чого проходить розділення даних на навчальну та тестову та виконується безпосереднє тренування моделі за допомогою нижче представленого коду.

```
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,
    warmup_steps=50,
    weight_decay=0.01,
    evaluation_strategy="steps",
    eval_steps=100,
    logging_steps=100,
    logging_dir="./logs",
    save_steps=500,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
```

Для тренування налаштовується кількість епох, розмір батчу, кількість кроків розігріву тощо. Виклик `trainer.train()` запустить повний цикл навчання, автоматично виконуючи прямий і зворотний проходи моделі, логуючи прогрес і зберігаючи чекпойнти

5.2.2 HMM-моделі

Приховані марковські моделі (Hidden Markov Models, HMM) належать до ймовірнісних підходів, які дають змогу описувати послідовності спостережень за

допомогою набору схованих станів і матриць імовірностей переходів. У контексті класифікації діалогів HMM дозволяють враховувати ланцюгову структуру реплік та відстежувати, як імовірність перебування в певному сервісі або інтенції змінюється з часом.

У марковській моделі вважається, що кожен стан залежить від попереднього зі сталою ймовірністю переходу. У HMM додатково передбачається, що справжні стани є прихованими, а спостереження, наприклад репліки користувача або ознаки тексту, лише опосередковано свідчать про те, у якому стані перебуває система. Для опису використовується матриця переходів A, де a_{ij} означає ймовірність перейти зі стану i в стан j, матриця емісій, яка показує, наскільки імовірно побачити те чи інше спостереження, наприклад, певне слово чи фразу в в кожному схованому стані та початкові імовірності π, що вказують, із якою ймовірністю система перебуває у тому чи іншому стані на початку.

Для класифікації в межах одного діалогу можна припустити, що кожен сервіс або інтенція є схованим станом, а фактичні репліки користувача чи системи – спостереженнями. У кожній репліці нам потрібно з’ясувати, яким є поточний стан: якщо ланцюг послідовності сервісів почався з готелю, то ймовірність залишитися у стані готелю на наступному кроці може бути вищою за різкі зміни до рестарану або потягу. Таким чином, HMM надає механізм поетапної оцінки того, які сервіси переважають у діалозі та як вони переходять один в один. На рисунку 5.2 зображена блок схема марковської моделі.

Стан означає внутрішнє положення системи, наприклад, активний сервіс, тоді як емісійні ймовірності вказують, із якою ймовірністю з такого стану може з’явитися конкретне спостереження, наприклад, слова «room», «booking», «I want a reservation». Ці ймовірності оцінюються під час тренування за допомогою алгоритма Баума-Велша, і далі в процесі класифікації застосовують алгоритм Вітербі, який допомагає знайти найбільш імовірний ланцюг схованих станів для заданої послідовності реплік.

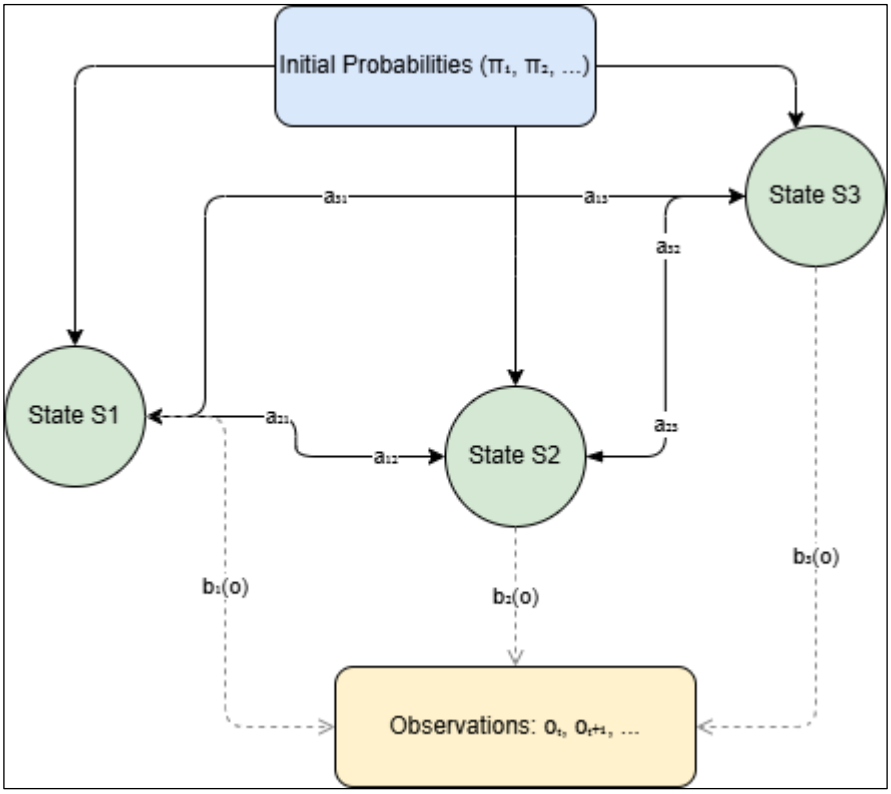

Рисунок 5.2 – Блок схема марковської моделі (рисунок виконаний самостійно)

Оскільки HMM можна застосовувати до задач, де послідовність реплік має вирішальне значення, її зручно використовувати в діалогових системах із визначеною логікою переходів між станами.

5.2.2.1 Стандартна HMM-класифікація

Для побудови марковської моделі (HMM) для класифікації реплік за сервісами у діалогах, додатково використовувалося TF-IDF для векторизації тексту та реалізація GaussianHMM із бібліотеки hmmlearn. TF-IDF (Term Frequency – Inverse Document Frequency) було обрано, оскільки він є методом перетворення тексту в числові вектори, який знаходить значущі слова, враховуючи їхню частоту в межах кожного документу та навпаки – рідкісність у корпусі загалом. Такий спосіб підходить для коротких реплік, характерних для діалогів: загальні фбо частовживані слова отримують нижчу вагу, а специфічні до сервісу (наприклад, «booking» для «hotel» чи «schedule» для «train») мають вищу вагу, що полегшує HMM розрізняти ймовірні стани.

У свою чергу GaussianHMM є конкретним різновидом HMM, що передбачає, що емісії розподілені за багатовимірним нормальним розподілом у кожному прихованому стані. Це дозволяє алгоритму оцінювати середнє значення та коваріаційну матрицю для кожного стану, отже – описувати, наскільки скупчено або розсіяно розташовані ознаки в просторі TF-IDF. Кожен стан, який відповідає певному сервісу, матиме свої параметри Гаусів, що полегшує імовірнісне визначення близості слів у векторному просторі. Код, який відповідає за навчання, представлений нижче.

```
    def train_and_save_model(X, lengths, all_service_labels, params, output_folder):
        n_components = params["n_components"]
        n_iter = params["n_iter"]
        covariance_type = params["covariance_type"]

        model = GaussianHMM(n_components=n_components,
                            covariance_type=covariance_type,
                            n_iter=n_iter,
                            random_state=42,
                            verbose=True)
        model.fit(X, lengths)
        state_mapping = compute_state_mapping(model, X, all_service_labels, n_components)

        os.makedirs(output_folder, exist_ok=True)
        output_filename = f"hmm_ncomp{n_components}_niter{n_iter}_{covariance_type}.pkl"
        output_path = os.path.join(output_folder, output_filename)
        save_model(model, vectorizer, state_mapping, output_path)
```

Для попередньої обробки тексту використовується код, що наведений у додатку Б. Після цього усі завантажені репліки збираються в один масив для векторизації за допомогою TF-IDF. За це відповідає функція vectorize_utterances, використання якої зображено нижче.

```
    X, lengths, vectorizer = vectorize_utterances(all_utts, dialogue_utterances)
```

Функція `vectorize_utterances` ініціалізує `TfidfVectorizer`, викликає `vectorizer.fit(all_utts)`, щоб зібрати словник і обчислити ваги IDF, а потім для кожного діалогу перетворює всі його репліки в матрицю ознак (TF-IDF).

Отримані вектори об’єднуються в один великий масив *X*, а довжини діалогів записуються в *lengths*, щоб HMM знала, де закінчується одна послідовність і починається інша. Далі викликається функція `train_and_save_model` для навчання і збереження моделі. Останнім кроком є визначення того, який сервіс найчастіше відповідає кожному латентному стану. Це робиться за допомогою функції:

```
    def compute_state_mapping(model, X, true_labels, n_components):
        predicted_states = model.predict(X)
        state_labels = defaultdict(list)
        for state, label in zip(predicted_states, true_labels):
            state_labels[state].append(label)
        mapping = {}
        for state in range(n_components):
            if state_labels[state]:
                mapping[state]                                          =
Counter(state_labels[state]).most_common(1)[0][0]
            else:
                mapping[state] = "none"
        return mapping
```

В цій функції для кожного рядка ми отримуємо набір станів, після чого перевіряємо, які сервіси частіше за все трапляються у вибірках із цим станом. У результаті кожному стану присвоюється мітка і в майбутньому при розпізнаванні можна буде інтерпретувати стан моделі як конкретний сервіс.

5.2.2.2 HMM з оптимізацією SVD

Для оптимізації ми використали TruncatedSVD, аби зменшити розмірність векторів після перетворення реплік у TF-IDF. TF-IDF генерує високу кількість ознак, адже враховує наявність тисяч унікальних слів чи підслів у діалогах. Завдяки SVD ми звужуємо цей простір до обмеженої кількості компонент, що зберігають найбільш суттєві патерни, тоді як малопоширені або випадкові особливості відсікаються. У результаті GaussianHMM має справу з коротшими, проте більш змістовними векторами, що прискорює процес оптимізації та полегшує пошук імовірнісних переходів між латентними станами. Приклад використання представлений кодом нижче.

```
def reduce_dimensionality(X, n_components=100):
    svd = TruncatedSVD(n_components=n_components, random_state=42)
    X_reduced = svd.fit_transform(X)
    return X_reduced, svd
```

У коді спочатку формується матриця TF-IDF на основі всіх реплік, а потім вона надходить до функції `reduce_dimensionality`, яка виконує `TruncatedSVD(n_components=n_components_svd)`. Під час цього кроку найбільш значущі компоненти зберігаються, а зайві виміри ігноруються. Після SVD слова, семантично близькі, утворюють подібні координати в новому просторі. Для `GaussianHMM` це означає, що емісійні ймовірності розраховуватимуться вже не на величезному розмірному просторі, а на стислому, де краще видно відмінності між репліками, притаманними різним сервісам.

Інший код залишається таким самим, який був у варіанті без оптимізації. Єдина відмінність полягає в тому, що `model.fit(X_reduced, lengths)` викликається зі зменшеною розмірністю. Так HMM навчається визначати матрицю переходів між латентними станами, а також гаусові розподіли ознак усередині кожного стану. Це спрощує завдання, оскільки менша кількість ознак означає нижчу складність для EM-алгоритму. У випадку GaussianHMM використовується алгоритм Baum–Welch для знаходження таких параметрів, які максимізують правдоподібність даних, навіть якщо реальний стан системи або розподіл не спостерігаються безпосередньо.

5.2.3 Наївний баєсівський класифікатор

Для класифікації діалогів за допомогою баєсівського класифікатора використовується підхід, який базується на класичному конвеєрі, який поєднує векторизацію текстів методом TF-IDF, багатоміткове кодування міток за допомогою MultiLabelBinarizer та застосування алгоритму Multinomial Naive Bayes у рамках One-vs-Rest класифікації.

MultiLabelBinarizer із бібліотеки scikit-learn перетворює вихідні мітки, які можуть бути представлені як список класів для кожного зразка, у двійкову матрицю. Кожен стовпець цієї матриці відповідає конкретній мітці, а кожен рядок

містить 1, якщо дана мітка присутня, або 0 – якщо ні. Це дозволяє подати дані для багатоміткової класифікації у вигляді числового масиву, зручного для обробки алгоритмами машинного навчання.

Алгоритм Multinomial Naive Bayes є імовірнісним класифікатором, який особливо добре працює з текстовими даними. Він оперує частотними ознаками або TF-IDF коефіцієнтами, що робить його ефективним для обробки даних, де ознаки (слова) мають дискретний розподіл. Основне припущення Multinomial Naive Bayes полягає в умовній незалежності ознак, що значно спрощує обчислення й дозволяє швидко навчатися на великих обсягах даних.

Щоб адаптувати Multinomial Naive Bayes до задачі, де один зразок може належати до кількох класів одночасно, застосовується підхід One-vs-Rest. У цьому підході для кожного класу будується окремий бінарний класифікатор, який відокремлює зразки, що належать до цього класу, від тих, що не належать. Потім результати всіх цих класифікаторів комбінуються, що дозволяє моделі робити багатоміткові передбачення. Таким чином, кожен вхідний текст може бути віднесений до декількох сервісів, якщо це потрібно, або до одного класу в разі однозначної класифікації.

Поєднання MultiLabelBinarizer, TF-IDF векторизації та алгоритму Multinomial Naive Bayes у рамках підходу One-vs-Rest створює потужний конвеєр для обробки текстових даних. TF-IDF дозволяє отримати числове представлення тексту, де часті, але менш значущі слова отримують нижчі ваги, а рідкісні та характерні слова мають вищу значущість. Це сприяє тому, що класифікатор може ефективніше визначати, які слова є індикаторами конкретних сервісів чи інтенцій. Такий підхід забезпечує швидке навчання та прийнятну точність, що робить його корисним як для попереднього аналізу, так і для інтеграції у більш складні системи діалогової класифікації.

Для реалізації описаного підходу спочатку підготовлюються дані за допомогою коду, який наведено у додатку В. Після цього створюється об'єкт MultiLabelBinarizer, який кодує мітки у двійковий формат. За допомогою виклику:

```
mlb = MultiLabelBinarizer(classes=sorted(allowed_labels))
Y_utter = mlb.fit_transform(utterance_labels)
Y_dialog = mlb.transform(dialog_labels)
```

Спочатку клас MultiLabelBinarizer ініціалізується із списком дозволених міток, відсортованих для узгодженості. Потім метод `fit_transform` перетворює мітки для реплік у вигляді двійкової матриці, де кожен стовпець відповідає конкретній мітці, а рядки містять 1 або 0 залежно від наявності мітки. Для діалогів використовується метод `transform`, який перетворює діалогові мітки у схожий формат.

Наступним кроком створюються два конвеєри за допомогою функції make_pipeline. Ці конвеєри складаються з TF-IDF векторизатора, який перетворює текстові дані у числові вектори, і класифікатора One-vs-Rest, заснованого на алгоритмі Multinomial Naive Bayes. Перший конвеєр використовується для класифікації окремих реплік, а другий – для класифікації цілих діалогів.

```
    pipeline_utter = make_pipeline(TfidfVectorizer(),
OneVsRestClassifier(MultinomialNB()))
    pipeline_dialog = make_pipeline(TfidfVectorizer(),
OneVsRestClassifier(MultinomialNB()))
```

Для навчання конвеєра застосовуємо метод fit на тренувальних даних. Для класифікації на рівні діалогу використовується аналогічний підхід, де весь текст діалогу подається у відповідний конвеєр.

```
pipeline_utter.fit(X_train, Y_train)
pipeline_dialog.fit(dialog_texts, Y_dialog)
```

Таким чином, даний код реалізує багатоміткову класифікацію як на рівні окремих реплік, так і на рівні всього діалогу, поєднуючи ефективну векторизацію тексту (TF-IDF) та швидкий, простий алгоритм Multinomial Naive Bayes у форматі One-vs-Rest.

5.3 Мультимодальні моделі

Мультимодальні підходи дають змогу одночасно враховувати різні типи даних, наприклад текст та зображення, і будувати більш краще розуміння змісту. Однією з відомих архітектур, що реалізують цей принцип, є CLIP (Contrastive Language-Image Pre-training). Ключова ідея CLIP полягає в тому, щоб навчити спільний простір ознак для тексту і зображень, аби надалі можна було зіставляти текстові та візуальні дані без додаткового навчання.

На рисунку 5.3 показано як відбувається попереднє навчання моделі. Зображення подаються на вхід Image Encoder, а текст – на вхід Text Encoder. На виході обидва енкодери генерують вектори: *I1*, *I2*, … для зображень і *T1*,*T2*,… для текстів.

Унікальною властивістю CLIP є її здатність до нульового навчання (zero-shot learning), що дозволяє використовувати модель для класифікації нових класів без потреби додаткового донавчання. Це відкриває широкі можливості для застосування в реальних умовах, де неможливо заздалегідь підготувати навчальні приклади для кожної категорії.

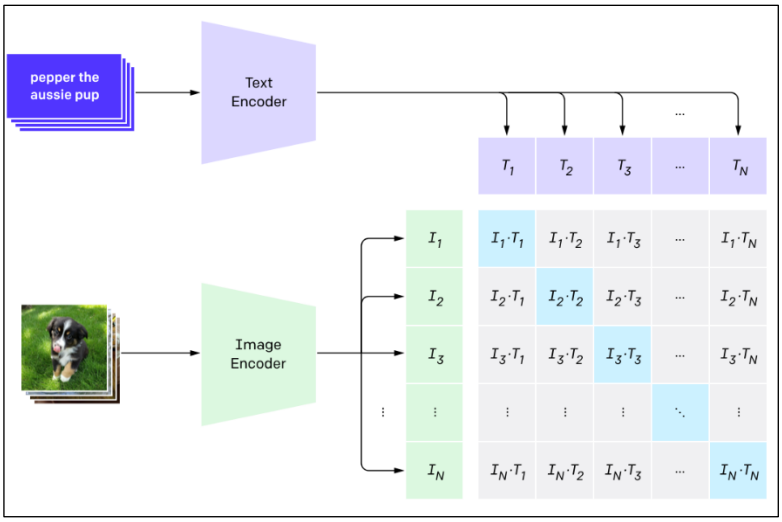

Рисунок 5.3 – Навчання CLIP моделі (https://openai.com/index/clip/)

Функція втрат налаштовує модель так, щоб вектор тексту був найбільш близьким до вектора відповідного зображення і найменш близьким – до всіх

інших. Таким чином, CLIP навчається встановлювати відповідність між зображенням і його текстовим описом.

Модель CLIP може використовуватися для задач у режимі нульового навчання. Цей процес схематично зображено на рисунку 5.4.

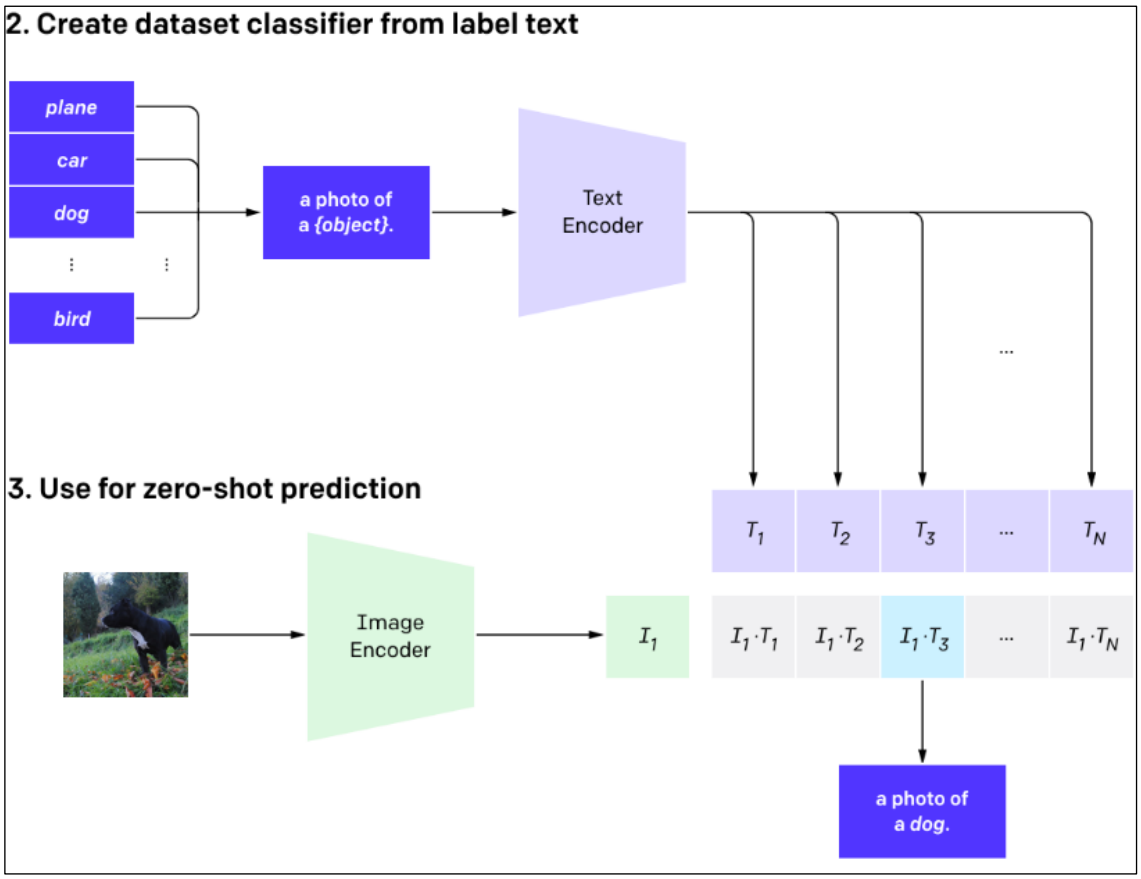

Рисунок 5.4 – Приклад використання моделі CLIP (https://openai.com/index/clip/)

Спочатку формуємо текстовий опис класів. Text Encoder перетворює ці описи у вектори $T1,T2,\ldots, T_n$. Потім, коли подається нове зображення, Image Encoder генерує його вектор I. Порівнюючи I з кожним T_k, отримуємо оцінки схожості, за якими можна визначити, якому класу належить це зображення.

Для реалізації моделі CLIP був створений клас CLIPClassifier. Архітектура моделі CLIPClassifier побудована на попередньо завантаженій моделі CLIP, яка відповідає за отримання векторних представлень як для зображень, так і для тексту. Всі параметри моделі CLIP заморожуються, що дозволяє зосередитися на навчанні лише додаткового класифікаційного шару, який приймає конкатеновані вектори ознак і видає відповідно класифікацію.

```
class CLIPClassifier(nn.Module):
    def __init__(self, clip_model, feature_dim, num_classes):
        super(CLIPClassifier, self).__init__()
        self.clip_model = clip_model
        for param in self.clip_model.parameters():
            param.requires_grad = False
        self.classifier = nn.Linear(feature_dim * 2, num_classes)

    def forward(self, image, text):
        image_features = self.clip_model.encode_image(image)
        text_features = self.clip_model.encode_text(text)
        image_features = image_features / image_features.norm(dim=-1,
keepdim=True)
        text_features  =  text_features  /  text_features.norm(dim=-1,
keepdim=True)
        combined = torch.cat([image_features, text_features], dim=1)
        logits = self.classifier(combined)
        return logits
```

Таким чином, завдяки описаній архітектурі CLIP модель (CLIPClassifier) може не лише розпізнавати об’єкти на зображеннях чи розуміти семантику тексту, а й поєднувати обидва параметра повідомлення для покращеного аналізу. Зокрема, під час обробки діалогів, де користувач може надсилати як текстові повідомлення, так і фотографії, така модель дає змогу одночасно сприймати контент повідомлення й розпізнавати візуальні вкладені медіа файли.

5.4 Аналіз отриманих результатів

Під час аналізу отриманих моделей ми вимірювали ключові показники, які дозволяють оцінити якість роботи класифікатора як на рівні окремих ходів, так і на рівні цілих діалогів. Основними метриками були Precision, Recall, F1-score та Accuracy.

Precision визначається як відношення кількості правильно передбачених позитивних зразків (TP) до загальної кількості зразків, які модель позначила як позитивні (TP + FP). Це відношення виражено у формулі 5.1

$$Precision = \frac{TP}{TP+FP} \tag{5.1}$$

Цей показник характеризує, наскільки модель точна у своїх позитивних передбаченнях.

Recall показує, яку частку від загальної кількості істинних позитивних зразків (TP + FN) модель змогла правильно знайти. Цей показник рахується у формулі 5.2.

$$Recall = \frac{TP}{TP+FN} \tag{5.2}$$

Recall допомагає зрозуміти, чи не пропускає модель позитивні випадки.

F1-score – це гармонійне середнє Precision та Recall, яке забезпечує збалансовану оцінку продуктивності, особливо при незбалансованих даних. Обчислюється за формулою 5.3.

$$F1 = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision} + Recal} \tag{5.3}$$

Accuracy відображає загальну частку правильно класифікованих зразків відносно всіх зразків та виражається формулою 5.4.

$$Accuracy = \frac{\text{T}P+TN}{\text{TP+TN+F}P+FN} \tag{5.4}$$

Окрім цих показників, ми обчислювали макро-середні метрики (macro-precision, macro-recall, macro-F1), які розраховуються як середнє арифметичне відповідних метрик по всіх класах. Це дозволяє оцінити загальну ефективність моделі незалежно від кількості зразків у кожному класі.

У задачах багатозначної класифікації кожен зразок може належати до декількох класів одночасно, тому істинні та передбачувані мітки представлені у вигляді багатовимірних векторів. Для кожного класу обчислюються окремі TP, FP, FN та TN, після чого метрики розраховуються так само, як для бінарної

класифікації. Далі ці метрики можуть бути агреговані за допомогою двох основних підходів, а саме:

- Macro-average**:** Обчислюється метрика для кожного класу окремо, після чого береться їх середнє арифметичне. Це підходить для оцінювання моделі, коли кожен клас має однакову важливість, незалежно від його представленості у даних;
- Micro-average**:** Спочатку підсумовуються всі TP, FP, FN і TN по всіх класах, а потім на основі цих сум обчислюються метрики. Цей підхід більше враховує кількість зразків і добре підходить для незбалансованих даних.

Ці показники дозволяють комплексно оцінити, наскільки добре модель справляється із завданням класифікації, виявити сильні та слабкі сторони роботи моделі, а також вказують на області, де потрібне подальше вдосконалення.

5.4.1 Модель BERT

Під час аналізу отриманих показників для моделі BERT, представлених на зображеннях, можна відзначити, що загальна точність (див. рис. 5) моделі досить висока (більше 90%) для найбільш популярних класів, що свідчить про здатність BERT правильно класифікувати зразки, які мають чітко визначені семантичні ознаки. Однак для деяких класів спостерігається нижчий рівень Recall, що свідчить про те, що модель не завжди встигає правильно розпізнати всі істинні позитивні випадки, особливо коли дані мають неоднозначний контекст або низьку представленість.

Precision, який відображає точність позитивних передбачень, у показниках є досить стабільним для класів із великою кількістю зразків, тоді як для менш представлених категорій він може бути дещо нижчим. Це можна пояснити тим, що BERT, завдяки своїй трансформерній архітектурі, ефективно вилучає контекстну інформацію, але у випадках, коли семантичні межі між класами нечіткі або дані для деяких класів обмежені, модель може помилково класифікувати деякі зразки, що знижує загальну якість передбачень.

Значення F1-score, яке є гармонійним середнім Precision та Recall, демонструють збалансовану оцінку продуктивності моделі. Як видно з графічного представлення на рисунку 5.5, найбільш популярні класи досягають високих F1-score, що свідчить про їхню стабільну класифікацію. Проте для класів з меншим числом прикладів F1-score знижується, що, ймовірно, пов'язано з дисбалансом даних і відсутністю достатньої кількості навчальних прикладів для належного представлення всіх особливостей цих категорій.

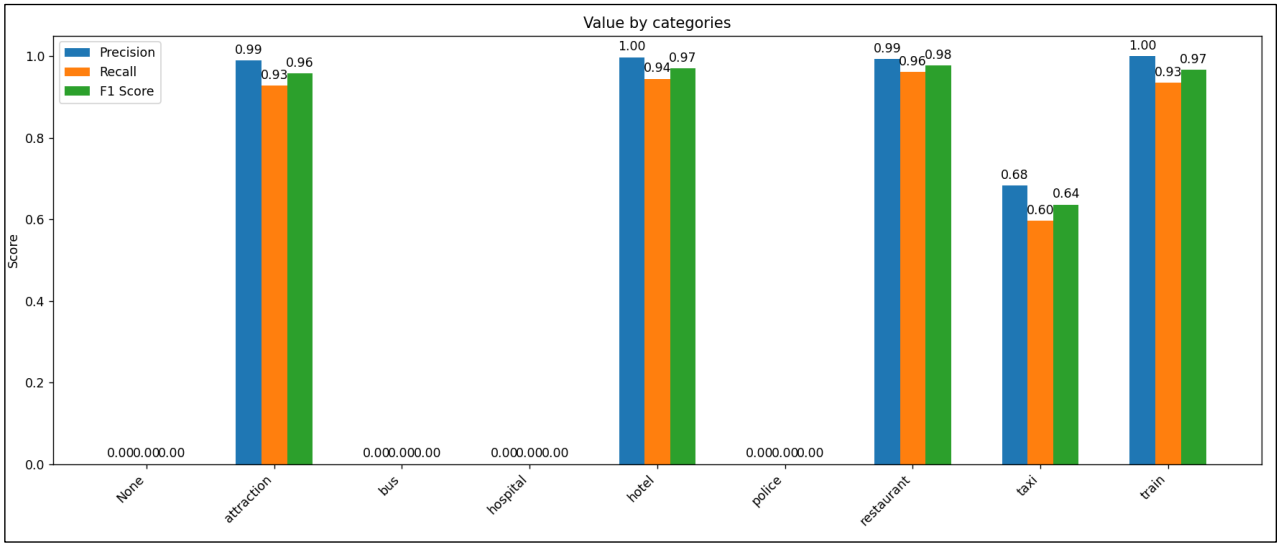

Рисунок 5.5 – Показники моделі BERT для окремих класів (рисунок виконаний самостійно)

У випадках багатозначної класифікації класифікації використовувалися макро-середні метрики. Результати зображені на рисунку 6. Макро-середнє дозволяє оцінити ефективність моделі, незалежно від розміру класів, шляхом обчислення середніх значень Precision, Recall та F1-score для кожного класу окремо. Це підкреслює, що незважаючи на високі показники для основних класів, менш представлені категорії можуть суттєво впливати на загальний баланс метрик.

Можливо, саме тому деякі макро-метрики виявилися нижчими, що свідчить про необхідність додаткового балансування даних або використання методів підсилення представлення рідкісних класів.

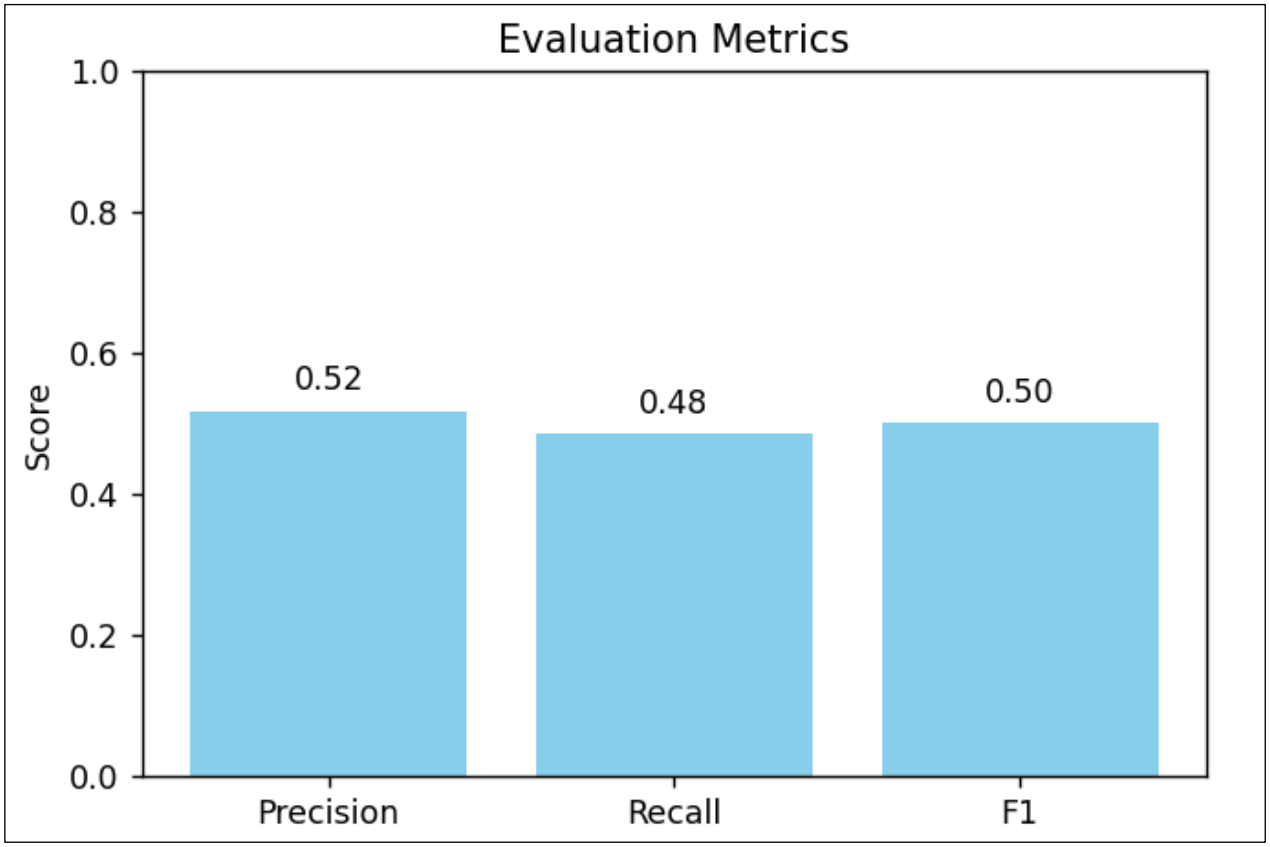

Рисунок 5.6 – Макро-середні метрики для моделі BERT

Таким чином, результати, представлені на зображеннях, свідчать про те, що модель BERT демонструє високий рівень загальної точності для добре представлених класів, проте має певні труднощі з розпізнаванням рідкісних або семантично неоднозначних категорій. Це може бути пов’язано як із специфікою даних, так і із характером моделі, яка, хоч і ефективно вилучає контекстну інформацію, проте залишається чутливою до дисбалансу в наборі даних.

5.4.2 Модель HMM

Під час аналізу отриманих результатів для моделі HMM було проведено серію експериментів із варіюванням основних параметрів, що впливають на здатність моделі адекватно описувати послідовність ходів у діалогах. Зокрема, досліджувались параметри: кількість компонент (ncomp) та параметр для зменшення розмірності ознак методом SVD (svd). Результати представлені у таблиці 5.1.

Таблиця 5.1 – Результати класифікації за допомогою HMM

Group	Dialogue Acc (%)	Precision	Recall	F1
ncomp4.0-svd100.0	53.9	0.18	0.23	0.20
ncomp4.0-svd1000.0	43.7	0.12	0.23	0.14
ncomp6.0-svd100.0	56.0	0.24	0.25	0.17
ncomp6.0-svd1000.0	52.5	0.20	0.25	0.21
ncomp8.0-svd100.0	63.0	0.23	0.29	0.21

Параметр ncomp визначає кількість латентних станів, які використовуються для моделювання внутрішньої структури даних. При збільшенні ncomp від 4 до 8 модель отримує можливість більш тонко відображати семантичні нюанси діалогу. Це відобразилося в тому, що діалогова точність зросла від 53–56% при ncomp = 4 або 6 до 63% при ncomp = 8.

Параметр svd впливає на зменшення розмірності простору ознак, що отримуються з TF-IDF векторів. Використання svd = 100 дозволяє зменшити шум і уникнути перенавчання, проте може призводити до втрати важливої інформації. З іншого боку, використання svd = 1000 зберігає більше інформації, але іноді зберігає і зайвий шум, що може негативно впливати на точність передбачень.

Результати експериментів свідчать, що при відносно низькій per-turn accuracy (близько 29–37%) агрегування передбачень за допомогою majority vote на рівні діалогу дозволяє досягти значно вищої точності (до 63%). Це говорить про те, що хоча окремі ходи можуть класифікуватися з помилками, узагальнення інформації в межах одного діалогу сприяє кращому визначенню загальної категорії. Також варто відзначити, що збільшення кількості компонент (ncomp) сприяє підвищенню діалогової точності, що можна пояснити кращим вилученням тонких семантичних особливостей тексту.

5.4.3 Модель CLIP

Під час аналізу моделі CLIP було проведено низку експериментів, що мали на меті визначити, наскільки ефективно вона здатна обробляти текстові та візуальні дані одночасно. У дослідженні використовувався датасет із зображеннями (постерами фільмів) та відповідними текстовими описами (сюжет, жанри тощо). Це дозволило перевірити здатність CLIP моделювати семантичні зв’язки між візуальною та текстовою інформацією. Результати надані у таблиці 5.2.

Таблиця 5.2 – Результати роботи моделі CLIP

Genre	Precision	Recall	F1
Action	0.7575	0.4872	0.5938
Biography	0.8735	0.6557	0.7491
Comedy	0.6058	0.7253	0.6606
Crime	0.7309	0.5977	0.6577
Documentary	0.5943	0.7046	0.6443
Drama	0.5392	0.8396	0.6551
Family	0.7448	0.6359	0.6865
Horror	0.7569	0.6572	0.7035
Music	0.8028	0.4956	0.6148
Mystery	0.5760	0.4253	0.4891
Talk-Show	0.0000	0.0000	0.0000
Thriller	0.7728	0.6802	0.7239
War	0.6690	0.6690	0.6690

Ця таблиця дає змогу оцінити, наскільки якісно модель розпізнає різні жанри за показниками Precision, Recall та F1. Деякі жанри, наприклад Biography (Precision = 0.8735, Recall = 0.6557, F1 = 0.7491) та Romance (0.7633, 0.7349,

0.7489), мають досить збалансовані показники й вирізняються високим рівнем точності й повноти.

Інші жанри, наприклад Animation (0.8045, 0.6170, 0.6994) та Horror (0.7569, 0.6572, 0.7035), також демонструють хороші результати, хоча Recall у них дещо нижчий за Precision. Протилежна ситуація спостерігається для Drama (0.5392, 0.8396, 0.6551), де висока повнота компенсує низьку точність, у підсумку даючи середнє F1.

Деякі категорії – News, Reality-TV та Talk-Show – отримали нульові показники. Швидше за все, тут немає достатньо прикладів (або вони зовсім відсутні) для навчання моделі, тож результати вийшли рівними нулю.

Жанр Short (0.5000, 0.1808, 0.2654) має високу точність, але надто низький показник Recall, через що F1-score досить малий. Це свідчить про те, що модель часто правильно класифікує короткометражні фільми, коли робить таке передбачення, але пропускає більшість із них, не розпізнаючи як Short.

Висока Precision означає, що модель рідко помиляється, відносячи стрічки до певного жанру, а висока Recall говорить про вміння знаходити всі фільми цього жанру серед решти.

5.4.3 Аналіз наївного баєсівського класифікатору

Аналізуючи результати роботи наївного баєсівського класифікатору видно, що модель класифікує діалоги досить успішно для деяких категорій, але зовсім не виявляє або ігнорує інші. Зокрема, добре помітно, що для класів hotel, restaurant та train модель досягає дуже високих значень Precision та Recall, майже не припускаючись помилок під час визначення цих категорій. Результати предсьавлены в таблиці 5.3.

У той же час, класи bus, hospital і police відсутні у передбаченнях моделі: Precision і Recall дорівнюють нулю, що свідчить або про брак подібних зразків у тестовому наборі, або про повну нездатність моделі виявляти такі діалоги. Клас taxi також є проблемним, адже при максимальній точності (Precision = 1.0) модель має вкрай низький показник Recall, тобто вона правильно визначає діалоги про

таксі лише тоді, коли робить таке передбачення, але при цьому пропускає переважну більшість істинних прикладів

Таблиця 5.3 – Результати роботи баєсівського класифікатору

Label	Precision	Recall	F1-Score
attraction	0.984556	0.639098	0.775076
bus	0.000000	0.000000	0.000000
hospital	0.000000	0.000000	0.000000
hotel	0.997340	0.949367	0.972763
police	0.000000	0.000000	0.000000
restaurant	0.973171	0.896629	0.933333
taxi	1.000000	0.121212	0.216216
train	1.000000	0.936065	0.967265
micro avg	0.989522	0.784528	0.875181

Це вказує на те, що модель правильно ідентифікує ті зразки, які визнає належними до цього класу, проте не розрізняє частину істинних позитивів.

6 ВИСНОВКИ

Під час дослідження було здійснено аналіз ефективності сучасних підходів до класифікації текстових повідомлень. До розгляду увійшли трансформери (BERT), мультимодальні моделі (CLIP), HMM, а також класичні алгоритми, такі як Softmax Regression, Naive Bayes, SVM, Random Forest і KNN. Критерії порівняння включали кількість параметрів, здатність враховувати контекст, обсяг необхідної пам’яті, розмір словника, масштабованість та рівень інтерпретованості. Отримані результати дозволяють зробити наступні висновки.

Трансформери, зокрема BERT, продемонстрували найвищі показники за критеріями точності, масштабованості та врахування контексту. Модель BERT має 110 мільйонів параметрів, здатна обробляти до 512 токенів контексту та забезпечує масштабованість 6.35, що дозволяє ефективно працювати з великими наборами даних. Аналогічно, мультимодальні моделі, такі як CLIP, з 112 мільйонами параметрів і масштабованістю 5.33, виявилися надзвичайно ефективними у задачах інтеграції тексту й візуальних даних. Ці підходи є ідеальними для складних задач із високими вимогами до точності, таких як класифікація тексту в реальному часі чи аналіз багатомодальних даних.

HMM посіла середню позицію за загальною ефективністю. Її здатність враховувати послідовності даних та працювати з ймовірнісною структурою забезпечує певну конкурентоспроможність у задачах аналізу тексту, особливо для послідовностей. Модель має 50000 параметрів і враховує до 512 токенів контексту, хоча її масштабованість становить лише 1.60, що обмежує її продуктивність при роботі з великими обсягами даних.

Softmax Regression і Naive Bayes показали низькі результати. Ці моделі мають високу інтерпретованість, що дозволяє легко аналізувати їхні рішення. Однак їхня масштабованість (2.96 і 4.44 відповідно) та нездатність враховувати контекст роблять їх придатними лише для простих задач із невеликими наборами даних.

SVM, Random Forest і KNN виявилися найменш ефективними через низьку масштабованість (2.00, 3.57 та 1.28 відповідно) і відсутність здатності

враховувати контекст. Random Forest і KNN додатково страждають через високу ресурсоємність, особливо в умовах великих наборів даних.

Таким чином, на основі отриманих результатів трансформери (BERT), мультимодальні моделі (CLIP) та HMM є найкращими кандидатами для подальших досліджень, оскільки вони забезпечують оптимальний баланс між точністю, врахуванням контексту, масштабованістю та ефективністю використання ресурсів. Ці моделі здатні забезпечити високу продуктивність у складних сценаріях класифікації текстових повідомлень, тоді як моделі з низькими результатами, такі як Softmax Regression, Naive Bayes, SVM, Random Forest і KNN, доцільно виключити через їхні обмеження.

Під час практичного аналізу, проведеного на реальних наборах даних, висновки загалом підтвердили попередню оцінку. Зокрема, трансформери (BERT) і мультимодальні моделі (CLIP) продемонстрували не лише найвищу теоретичну конкурентоспроможність (з точки зору здатності враховувати контекст, масштабованості тощо), а й фактичну перевагу у точності й стабільності результатів. Модель BERT отримала найбільші значення Recall і F1-score на різноманітних текстових наборах, у той час як CLIP виявилася особливо корисною в задачах, де текст необхідно поєднувати з візуальними ознаками (наприклад, аналіз описів і зображень одночасно).

HMM підтвердила свою посередню позицію й на практиці: її результати були кращими за прості класифікатори на основі статистичних моделей, проте відчутно відставали від трансформерів, особливо на розширених масивах тексту з великою кількістю токенів. Незважаючи на меншу кількість параметрів, HMM виявилася здатною опрацьовувати послідовності даних і зберігати адекватну точність, якщо обсяги датасету не надто великі. Утім, через обмежену масштабованість продуктивність моделей цього класу все ж не дорівнювала сучасним трансформерам. Результати тестування Naive Bayes підтвердили їхню непридатність до задач із великим обсягом даних та необхідністю аналізу складного контексту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., Polosukhin I "Attention Is All You Need"/Conference NIPS, 2023, 15 с.

2. Kahana A., Elisha O. "MESSAGENET: MESSAGE CLASSIFICATION USING NATURAL LANGUAGE PROCESSING AND META-DATA"/ICLR Conference, 2023, 9 с.

3. Han J., Li J., Sun A. "UFTR: A Unified Framework for Ticket Routing"/arxiv, 2020, URL: https://arxiv.org/abs/2003.00703v1. (дата звернення: 12.20.2024)

4. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. & Sutskever, I. "Learning Transferable Visual Models From Natural Language Supervision."/ arxiv, 2021, URL: https://arxiv.org/abs/2103.00020 (дата звернення: 12.20.2024)

5. Transformer (deep learning architecture). URL: https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture) (дата звернення 12.20.24)

6. Naive Bayes classifier. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier (дата звернення 12.18.2024

7. The KNN Algorithm – Explanation, Opportunities, Limitations). URL: https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations (дата звернення 12.20.2024)

8. Neukirchen C., Rigoll G. " Controlling the Complexity of HMM Systems by Regularization"/ NIPS, 1998, 7с

9. Сокорчук І.П. "Автоматизація повсякденних завдань викладачів за допомогою засобів штучного інтелекту: покращення робочого процесу та ефективності". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, с. 330–331.

10. Каук В.І. "Генеративний штучний інтелект – креативний помічник дизайнера". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, с. 283–294.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

9 Сокорчук І.П. "Автоматизація повсякденних завдань викладачів за допомогою засобів штучного інтелекту: покращення робочого процесу та ефективності". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, с. 330–331.

10 Каук В.І. "Генеративний штучний інтелект – креативний помічник дизайнера". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, с. 283–294.