Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Система авторської атрибуції із застосуванням ансамблю класичних і нейронних моделей
(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-24-1
Капін С.Д.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)
(повна назва спеціалізації)
Керівник доц. Волощук О.Б.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лариса ЧАЛА
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Науки про дані (Data Science)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Капіну Сергію Дем'яновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система авторської атрибуції із застосуванням ансамблю класичних і нейронних моделей

затверджена наказом університету від 24 листопада 20 25 р. № 1057Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 грудня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації та дані Інтернет-джерел, Python documentation, набір даних для навчання системи.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Проєктування системи

3) Реалізація програмного забезпечення

4) Експериментальні дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	24.11.2025	виконано
2	Аналіз предметної галузі	25.11.2025	виконано
3	Проєктування системи	27.11.2025	виконано
4	Реалізація програмного забезпечення	29.11.2025	виконано
5	Експериментальні дослідження	05.12.2025	виконано
6	Написання пояснювальної записки	08.12.2025	виконано
7	Перевірка на академічний плагіат	09.12.2025	виконано
8	Нормконтроль	10.12.2025	виконано
9	Підготовка презентації та доповіді	11.12.2025	виконано
10	Попередній захист	12.12.2025	виконано
11	Рецензування	13.12.2025	виконано
12	Захист перед ЕК	16.12.2025	виконано

Дата видачі завдання 24 листопада 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Волощук О.Б.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 131 с., 19 рис., 2 табл., 2 дод., 23 джерела.

АВТОРСЬКА АТРИБУЦІЯ, АНСАМБЛЕВА МОДЕЛЬ, ВЕРИФІКАЦІЯ АВТОРСТВА, СТИЛОМЕТРІЯ, MINILM, PYTHON, TELEGRAM, TF-IDF.

Об’єкт дослідження – процес авторської атрибуції коротких україномовних текстів у соціальних мережах і месенджерах.

Предмет дослідження – ансамблеві моделі та стилометричні ознаки для верифікації авторства коротких україномовних текстів.

Мета роботи – розроблення та експериментальне дослідження програмної системи авторської атрибуції коротких україномовних текстів із використанням ансамблю класичних та нейронних моделей.

Методи дослідження – аналіз наукових джерел з авторської атрибуції та машинного навчання, методи класичного машинного навчання (TF-IDF, логістична регресія, стилометричні ознаки), нейронні ембеддинги тексту на базі трансформерних моделей, експериментальні обчислювальні дослідження.

У роботі розроблено модульний Python-пакет authorship_attrib для авторської атрибуції коротких Telegram-повідомлень, який поєднує символьний TF-IDF, стилометричні ознаки та трансформерні ембеддинги в єдиному ансамблі. Експерименти на реальному корпусі показали, що багатокласова класифікація окремих повідомлень є нестійкою, натомість профільна верифікація з використанням ансамблю забезпечує надійне відокремлення текстів цільового автора та може слугувати основою для подальших прикладних рішень.

ABSTRACT

Master’s thesis contains: 131 pp., 19 fig., 2 tabl., 2 ann., 23 references.

AUTHORSHIP ATTRIBUTION, AUTHORSHIP VERIFICATION, ENSEMBLE MODEL, MINILM, PYTHON, STYLOMETRY, TELEGRAM, TF-IDF.

Object of the research – the process of authorship attribution of short Ukrainian-language texts in social networks and messengers.

Subject of the research – ensemble models and stylometric features for authorship verification of short Ukrainian-language texts.

Purpose of the work – the development and experimental study of a software system for authorship attribution of short Ukrainian-language texts using an ensemble of classical and neural models.

Methods of the research – analysis of scientific literature on authorship attribution and machine learning, classical machine learning methods (TF-IDF, logistic regression, stylometric features), neural text embeddings based on transformer models, and experimental computational studies.

The work presents a modular Python package, authorship_attrib, for authorship attribution of short Telegram messages, which combines character-level TF-IDF, stylometric features, and transformer-based embeddings into a single ensemble. Experiments on a real-world corpus showed that multiclass classification of individual messages is unreliable, whereas profile-based verification using the ensemble reliably distinguishes texts of the target author and can serve as a basis for further applied solutions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface – програмний інтерфейс застосунків;

BERT – Bidirectional Encoder Representations from Transformers – бінапрямні подання на основі трансформерів;

CLI – Command Line Interface – інтерфейс командного рядка;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

CPU – Central Processing Unit – центральний процесор;

CSR – Compressed Sparse Row – формат зберігання розріджених матриць зі стисненими рядками;

CSV – Comma-Separated Values – текстовий формат табличних даних зі значеннями, розділеними комами;

HTML – HyperText Markup Language – мова гіпертекстової розмітки;

JS – JavaScript – мова програмування JavaScript;

JSON – JavaScript Object Notation – текстовий формат обміну структурованими даними;

ML – Machine Learning – машинне навчання;

MiniLM – легка трансформерна модель для обчислення семантичних ембеддингів тексту;

NLP – Natural Language Processing – обробка природної мови;

SVM – Support Vector Machine – метод опорних векторів;

TF-IDF – Term Frequency – Inverse Document Frequency – статистична міра ваги термів у документі відносно корпусу;

URL – Uniform Resource Locator – уніфікований покажчик (адреса) ресурсу в Інтернеті;

YAML – YAML Ain’t Markup Language – формат текстових конфігураційних файлів.

ВСТУП

Сучасні соціальні мережі та месенджери генерують величезні обсяги коротких текстових повідомлень, значна частина яких є анонімною або псевдонімною. Це ускладнює виявлення фейкових акаунтів, координованих інформаційних кампаній та авторів потенційно шкідливого контенту, а також обмежує можливості цифрової криміналістики та аналітики безпеки. В умовах гібридних інформаційних загроз і масової комунікації в онлайн-середовищі особливої актуальності набувають методи авторської атрибуції, які дозволяють за стилем письма оцінювати належність тексту певному авторові.

Більшість існуючих підходів у стилометрії розроблені для відносно довгих, формальних текстів і слабо пристосовані до коротких, неформальних україномовних повідомлень із домішками сленгу, емодзі, змішаних мов і свідомих орфографічних відхилень. Класичні моделі на основі n-грам і простих статистичних ознак у таких умовах часто дають нестабільні результати, тоді як сучасні нейронні моделі потребують значних обсягів даних і обчислювальних ресурсів. Це створює потребу в практичних рішеннях, які поєднують переваги класичних стилометричних підходів і нейронних ембеддингів у єдиному ансамблі, придатному для реальних сценаріїв верифікації авторства.

Основною метою даної кваліфікаційної роботи є розробка та дослідження програмної системи авторської атрибуції коротких україномовних текстів із застосуванням ансамблю класичних стилометричних та нейронних моделей. Для досягнення цієї мети необхідно проаналізувати сучасні методи авторської атрибуції, обґрунтувати вибір ознак і моделей, реалізувати програмний інструментарій для побудови профілів авторів і порівняння текстів, а також провести експериментальне дослідження якості розробленої системи на корпусі реальних повідомлень соціальних мереж.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Постановка задачі та ключові виклики

Авторська атрибуція, також відома як стилометрія, є науковою дисципліною, що займається визначенням автора тексту на основі аналізу його стилю. В основі цього підходу лежить гіпотеза про те, що кожен автор володіє унікальним набором мовних звичок, які несвідомо проявляються в його роботах, формуючи своєрідний «літературний відбиток» [1], [2], [3], [4]. Ці патерни охоплюють вибір лексики, синтаксичних конструкцій, форматування та інші індивідуальні особливості.

Залежно від кінцевої мети, в авторській атрибуції виділяють кілька основних типів задач. Найпоширенішою є ідентифікація автора, що полягає у визначенні, хто з відомого переліку авторів-кандидатів написав анонімний текст. Іншим типом є кластеризація, де набір текстів групується за авторством без попередньої інформації про авторів. У цій кваліфікаційній роботі основна увага приділяється задачі верифікації авторства – бінарній класифікації, де необхідно встановити, чи були два тексти написані однією й тією самою людиною. На відміну від багатокласової ідентифікації автора, у цьому випадку головним є не визначення конкретного імені з фіксованого списку, а надійна оцінка стилістичної близькості між порівнюваними текстами.

Особливістю роботи є фокус не на великих літературних творах, а на коротких україномовних текстах, характерних для соціальних мереж та месенджерів. Практичний інтерес становлять насамперед повідомлення з публічних і приватних Telegram-каналів та чатів, де тексти є короткими, нерідко містять сленг, емодзі та інші шумові елементи. Це накладає суттєві обмеження на обсяг доступної інформації та стабільність стилометричних оцінок. Тому в рамках кваліфікаційної роботи ставиться задача розробити інструментарій авторської атрибуції, який поєднує класичні статистичні

підходи та нейромережеві моделі, формуючи ансамбль методів, стійкий до шуму та малих обсягів тексту.

1.2 Особливості коротких україномовних текстів соціальних мереж як об’єкта аналізу

Сучасне середовище, зокрема поширення соціальних мереж і месенджерів, ставить перед дослідниками авторської атрибуції низку складних викликів. Одним із головних є проблема коротких текстів. На відміну від художніх творів чи публіцистики, що надають великий обсяг матеріалу для статистичного аналізу, пости, коментарі та приватні повідомлення зазвичай містять від кількох десятків до кількох сотень символів. Це ускладнює одержання стабільних оцінок частот та робить будь-які стилометричні ознаки «шумними» й нестійкими [5].

Додаткову складність становить варіативність стилю самого автора. Той самий користувач може писати по-різному залежно від теми розмови, платформи, аудиторії чи емоційного стану. Для українськомовного сегмента інтернету характерні також явища код-світчингу (перемикання між українською, російською, англійською), суржику, використання латинки, нестандартної орфографії. Усе це знижує ефективність класичних інструментів обробки природної мови, які розраховані на більш формальний текст [6], [7].

Ще одна важлива риса – неформальний характер інтернет-комунікації: сленг, емодзі, хештеги, мемні конструкції, навмисні «криві» написання слів, надмірна пунктуація. Багато з цих елементів не підтримуються або погано інтерпретуються типовими NLP-пайплайнами, але при цьому можуть бути дуже інформативними з точки зору стилю. Нарешті, не можна виключати свідоме маскування стилю, коли автор намагається писати нехарактерно для себе, щоб зберегти анонімність чи заплутати аналіз. Це вимагає використання таких ознак, які складно

контролювати вручну, зокрема символьних n-грам, розподілів функційних слів та дрібних графічних патернів письма.

1.3 Стилометричні ознаки в авторській атрибуції

Для кількісного вимірювання авторського стилю використовуються різноманітні стилометричні ознаки. Історично першими та найбільш поширеними є лексичні ознаки, що аналізують словниковий запас автора. До них належать частота вживання певних слів, особливо функціональних (службових), оскільки їх вибір найменше залежить від тематики тексту. Також інформативними є показники багатства лексикону, наприклад співвідношення унікальних слів до загальної кількості (коефіцієнт type–token ratio), середня довжина слів і речень, частка різних частин мови [1], [8].

Згодом значного поширення набули символьні ознаки, які є більш стійкими до спроб свідомого маскування стилю, оскільки фіксують глибинні, підсвідомі патерни письма. Найбільш потужною ознакою цього класу є розподіл символьних n-грам – коротких послідовностей із N символів. Аналіз частоти появи біграм чи триграм дозволяє вловити унікальну «мелодику» тексту автора, незалежно від його змісту. До цієї ж категорії належить аналіз частоти використання розділових знаків, цифр та великих літер, а також специфічних комбінацій символів, пов’язаних із неформальним спілкуванням (смайли, повторювані знаки оклику тощо).

Більш абстрактними є синтаксичні ознаки, що описують структуру речень. Аналіз розподілу частин мови (іменників, дієслів, прикметників) або вимірювання складності синтаксичних дерев можуть дати уявлення про переваги автора у побудові фраз. Хоча ці ознаки теоретично є стійкими до зміни теми, їх практичне застосування ускладнюється необхідністю використання складних NLP-інструментів, таких як синтаксичні парсери, які часто дають збої на неформальних або коротких текстах.

Окрему групу складають ідіосинкратичні ознаки – унікальні звички автора, такі як характерні друкарські помилки, специфічні скорочення, манера використання емодзі, хештегів, оформлення абзаців тощо. В сукупності ці ознаки дозволяють сформувати багатовимірний профіль стилю автора, близький до концепції Writeprints.

У розроблюваній системі частина таких ознак використовується безпосередньо: символьні TF-IDF-n-грами, частоти пунктуації, емодзі, великої літери, розподіли функційних слів, середні довжини слів і речень. На їх основі формується густий стилометричний вектор, який відіграє роль writeprints-подібного профілю автора та надалі використовується як один з каналів ансамблю.

1.4 Аналіз методів та моделей

Розвиток методів авторської атрибуції пройшов шлях від простих статистичних підходів до складних нейромережевих архітектур. На ранніх етапах домінували відстаневі та частотні методи, які не потребували машинного навчання в сучасному розумінні. Згодом, із розвитком обчислювальних потужностей, перевагу почали отримувати методи класичного машинного навчання, що працюють із векторними поданнями тексту. Останніми роками до цього додалися нейромережеві підходи, побудовані на трансформерних моделях і семантичних ембеддингах.

З огляду на постановку задачі верифікації авторства для коротких неформальних текстів доцільно детальніше розглянути чотири групи підходів, які лягли в основу розроблюваної системи: статистичні відстаневі методи, методи класичного машинного навчання, writeprints-подібні багатофакторні профілі та нейромережеві моделі на основі семантичних ембеддингів.

1.4.1 Статистичні та відстаневі методи

Найпростішим підходом до авторської атрибуції є представлення тексту у вигляді вектора частот деякого набору ознак (як правило, найчастотніших слів чи функціональних одиниць), після чого обчислюється статистична «відстань» між профілями автора та анонімного тексту.

Класичний метод дельти Берроуза (Burrows' Delta) працює з найчастотнішими словами корпусу згідно з формулою (1.1) [9]. Для кожного слова у кожного автора та тексту розраховуються стандартизовані частоти (z-оцінки).

$$z_{AU,i} = \frac{f_{AU,i} - \mu_i}{\sigma_i}, \tag{1.1}$$

де $z_{A,i}$ та $z_{U,i}$ – стандартизована (z-нормалізована) відносна частота i-го слова для автора A або тексту U;

$f_{AU,i}$ – відносна частота i-го слова в текстах автора A або анонімному тексті U;

μ_i та σ_i – середнє значення та стандартне відхилення частоти цього слова по всіх авторах.

Відстань між автором A та анонімним текстом U обчислюється як середнє модульне відхилення згідно з формулою (1.2).

$$\Delta(A,U) = \frac{1}{m}\sum_{i=1}^{m} \mid z_{A,i} - z_{U,i} \mid, \tag{1.2}$$

де m – це кількість ознак, по яких рахується дельта.

Автор, для якого $\Delta(A,U)$ мінімальна, вважається найбільш ймовірним. У задачі верифікації авторства на основі Δ можна будувати порогове рішення: пари з $\Delta < \tau$ вважаються такими, що належать одному автору.

До цієї ж групи належать методи на основі класичних метричних відстаней у просторі частотних векторів – евклідова, мангеттенська, відстань Махаланобіса, а також інформаційно-теоретичні міри на кшталт дивергенції Кульбака-Лейблера або крос-ентропії між розподілами символів чи слів. У випадку символьних n-грам ці метрики добре працюють на коротких текстах, оскільки не залежать від семантики й чутливі до дрібних патернів написання (використання пробілів, розділових знаків, «нестандартних» емоційних конструкцій тощо) [10].

Статистичні методи авторської атрибуції мають кілька ключових переваг: простота реалізації, невисокі обчислювальні витрати, прозорість (легко прослідкувати, які саме слова чи n-грами впливають на відстань), відносна стійкість до змін тематики текстів. Водночас на дуже коротких текстах їх якість помітно погіршується через нестабільність оцінок частот; вони чутливі до домішок текстів інших авторів та різких стилістичних зрушень, а здатність моделювати складні нелінійні взаємозв’язки між ознаками є обмеженою.

У цій роботі метод дельти Берроуза використовується як окремий компонент ансамблю, що формує одну з базових оцінок подібності між текстами та, після калібрування, перетворюється на узагальнену ймовірнісну оцінку.

1.4.2 Методи класичного машинного навчання на основі n-грам

Більш гнучким підходом порівняно зі статистичними відстаневими методами є схема «векторизація тексту та подальша класифікація». Текст спочатку перетворюється у вектор фіксованої розмірності, а вже на цих векторах навчається модель машинного навчання. Найпоширеніший варіант – використання символьних або словних n-грам у поєднанні з TF-IDF-векторизацією. Інтуїтивно TF-IDF підвищує вагу тих n-грам, які часто зустрічаються в конкретному тексті, але при цьому є відносно рідкісними в

корпусі загалом. Для коротких і шумних текстів, характерних для соціальних мереж, особливо корисними виявляються символьні n-грами, оскільки вони чутливі до орфографічних звичок, пунктуації, емодзі та інших дрібних патернів написання, які важко підробити свідомо.

На отриманих векторах застосовуються стандартні алгоритми машинного навчання. Метод опорних векторів (SVM) добре працює з високорозмірними розрідженими векторами й часто є базовим вибором для текстової класифікації. Логістична регресія моделює ймовірність належності пари текстів до класу «один автор» і має приємний бонус у вигляді інтерпретованості ваг: за коефіцієнтами можна оцінити, які ознаки роблять внесок у позитивне чи негативне рішення. Ансамблеві методи на кшталт випадкового лісу або градієнтного бустингу будують композицію дерев рішень і часто дають високу якість «з коробки», хоча ціною цього є менша прозорість внутрішньої логіки моделі.

Окремий напрямок у межах класичного підходу представляє техніка Writeprints, яка задає великий, заздалегідь спроєктований набір стилометричних ознак, що описують текст з різних боків [11]. Частина ознак відображає лексику автора (розподіл довжин слів, співвідношення між функціональними та змістовими словами, частоти різних частин мови). Інша частина пов’язана з рівнем символів (частоти окремих символів, n-грам, розділових знаків, цифр, великих літер). Додаються синтаксичні характеристики (розподіл довжин речень, перевага певних граматичних конструкцій) та індивідуальні звички (типові помилки, скорочення, манера використовувати емодзі, оформлення списків тощо). У сумі це дає вектор з сотень або навіть тисяч компонент, який добре описує «почерк» автора.

Сильна сторона класичних методів – їхня гнучкість і помірні вимоги до ресурсів. Дослідник може комбінувати різні типи ознак – символьні, лексичні, синтаксичні – і відносно легко аналізувати важливість кожної з них. Водночас цей підхід вимагає ручної роботи: треба продумати, які ознаки збирати, як їх нормалізувати, які з них відсікати. За сильних змін

стилю або в доменах з обмеженими мовними ресурсами класичні моделі можуть деградувати.

У контексті цієї роботи модель на основі TF-IDF символьних n-грам доповнюється густим вектором ручних стилометричних ознак (частоти пунктуації, емодзі, великих літер, функційних слів, середні довжини тощо), який виконує роль writeprints-подібного профілю. Окремо для таких профілів обчислюється косинусна схожість. Обидва сигнали – TF-IDF-косинус і косинус густих стилометричних ознак – надалі використовуються в ансамблі разом із Delta та нейромережевими ембеддингами.

1.4.3 Нейромережеві підходи та моделі на основі ембеддингів

Нейромережеві методи пропонують іншу філософію розв’язання задачі: замість того, щоб вручну конструювати ознаки, дослідник делегує пошук корисних патернів самій моделі. Глибинні мережі отримують на вхід сирий текст у вигляді послідовності символів або токенів і в процесі навчання самостійно виявляють ті регулярності, які допомагають відрізняти одного автора від іншого.

Один із напрямів – символьно-рівневі CNN та RNN. У таких моделях текст розглядається як довга послідовність символів. Згорткові мережі зі вікнами невеликого розміру вловлюють локальні структури (типові поєднання букв, характерні емоційні послідовності знаків оклику й смайлів, нестандартне використання пробілів), а рекурентні мережі додають здатність запам’ятовувати контекст на довшому відрізку. Подібні моделі не залежать від граматично правильних речень чи повноцінних слів і можуть працювати навіть на дуже неформальному тексті [12].

Інший напрямок пов’язаний із словними та підсловними моделями. Там текст попередньо токенізується на слова або субслова, а кожен токен відображається у вектор у просторі ембеддингів (word2vec, fastText тощо). Далі поверх послідовності векторів будуються згорткові чи рекурентні

шари. Такі моделі сильніше орієнтуються на зміст, але за певних налаштувань можуть вловлювати і стилістичні особливості, наприклад схильність до певних конструкцій чи наборів слів.

Найпотужніший і зараз найпоширеніший підхід – використання трансформерних моделей. BERT-подібні архітектури, зокрема легкі моделі класу MiniLM, навчаються на великих корпусах і вміють перетворювати фрагмент тексту у щільний вектор фіксованої розмірності, який поєднує семантичну та частково стилістичну інформацію [13]. Для задачі верифікації авторства це дає простий сценарій: кожен текст перетворюється на ембеддинг, після чого оцінюється ступінь їх подібності, наприклад за косинусною мірою.

Більш просунутим варіантом є сіамські або триплетні архітектури, де дві (або три) копії моделі з розділеними вагами одночасно обробляють декілька текстів, а навчання відбувається за допомогою контрастивних функцій втрат, згідно з формулою (1.3).

$$L = y \cdot d^2 + (1 - y) \cdot \max(0, m - d)^2, \qquad (1.3)$$

де $y \in \{0,1\}$ – мітка (1 – той самий автор);

d – евклідова відстань між ембедиґами, m – відступ (margin).

Такий підхід явно навчає простір ембеддингів так, щоб тексти одного автора були ближчими один до одного, ніж до текстів інших авторів.

Перевага нейромережевих методів – здатність моделювати складні нелінійні взаємозв’язки, використовувати контекст і переносити знання з великих зовнішніх корпусів. Недоліки – потреба в значних обчислювальних ресурсах, схильність до перенавчання на малих даних, складність інтерпретації.

У рамках цієї роботи нейромережевий підхід використовується у відносно простій, але практичній формі: попередньо навчена трансформерна модель класу MiniLM (через бібліотеку sentence-

transformers) генерує семантичні ембеддинги для текстів, а косинусна подібність між цими ембеддингами розглядається як окремий сигнал «схожості стилю». Цей сигнал не замінює класичні методи, а доповнює їх, додаючи до ансамблю ще один незалежний «погляд» на дані.

1.4.4 Підходи до побудови моделей верифікації авторства

Задача верифікації авторства принципово відрізняється від класичної ідентифікації. Якщо в ідентифікації ми маємо фіксований набір авторів і намагаємося визначити, хто саме з них написав текст, то у верифікації питання ставиться інакше: чи належать два конкретні тексти одній і тій самій людині, незалежно від того, скільки всього авторів є в корпусі й чи бачила модель їх раніше [4]. Це вимагає інших алгоритмічних стратегій.

Один зі шляхів – профілювання авторів. Для кожного автора на основі його текстів формується узагальнений профіль: середній вектор TF-IDF, усереднений вектор стилометричних ознак або інший стислий опис стилю. Коли з’являється анонімний текст, його профіль порівнюється з профілями відомих авторів. У верифікації така стратегія працює, коли про автора накопичено достатньо матеріалу, але погано пристосована до випадків, коли текстів мало або автор є новим.

Більш прямим підходом є парна класифікація. Модель навчається не на окремих текстах, а на парах; кожна пара позначається міткою «один автор» / «різні автори». Для пари будуються ознаки, що описують їхню схожість: відстані, косинусні подібності, різниці ручних ознак, дивергенції розподілів. Далі на цих векторах навчається бінарний класифікатор (логістична регресія, SVM або невелика нейромережа), який видає ймовірність того, що тексти належать одному автору.

Нарешті, найбільш гнучким варіантом є ансамблеве комбінування, або мета-класифікація. Замість того щоб покладатися на один метод, будують кілька незалежних моделей, кожна з яких іншим способом оцінює

подібність між текстами. Наприклад, дельта Берроуза дає відстань у просторі частот найуживаніших слів, TF-IDF-модель – косинусну схожість у просторі символьних n-грам, writeprints-подібний профіль – окрему оцінку схожості стилометричних ознак, а трансформерна модель – косинусну подібність між ембеддингами. Додатково можуть використовуватися статистичні метрики на кшталт χ^2 та JS-дивергенції для розподілів функційних слів.

У цій роботі реалізовано саме ансамблевий підхід. Для кожної пари «профіль автора – цільовий текст» обчислюється вектор ознак, який включає:

- косинусну подібність у просторі TF-IDF символьних n-грам;
- за наявності ембеддингів – косинусну подібність у просторі MiniLM-ембеддингів;
- значення Delta та пов’язану з нею ймовірнісну оцінку;
- косинусну подібність між густими стилометричними (writeprints-подібними) векторами;
- X^2 та JS-дивергенцію розподілів функційних слів;
- абсолютні різниці ключових ручних стилометричних метрик.

На цьому багатовимірному векторі навчається лінійний мета-класифікатор на основі логістичної регресії, який виконує роль калібратора і повертає шукану ймовірність $p(\text{same\_author})$. Такий дизайн дозволяє одночасно використати як «класичні» стилометричні сигнали, так і потужність попередньо навчених нейромережевих моделей, при цьому зберігаючи відносну прозорість структури рішень.

1.5 Огляд існуючих програмних рішень

Для практичного застосування методів авторської атрибуції існує низка готових інструментів та бібліотек. Умовно їх можна поділити на

академічні пакети, орієнтовані на дослідників, та універсальні програмні бібліотеки.

До першої групи належить, наприклад, Stylo – потужний пакет для статистичного середовища R, що надає широкий функціонал для стилометричного аналізу, але вимагає від користувача навичок програмування [8]. Іншим відомим інструментом є JGAAP (Java Graphical Authorship Attribution Program) – автономний застосунок з графічним інтерфейсом, який дозволяє експериментувати з різними методами без написання коду [14]. Такі засоби зручні для попереднього аналізу та навчальних цілей, проте їх складніше інтегрувати у власні програмні комплекси та автоматизовані пайплайни.

Для розробки власних гнучких рішень найчастіше використовуються Python-бібліотеки. Універсальна бібліотека scikit-learn дозволяє реалізувати більшість класичних підходів: надає реалізації TF-IDF-векторизаторів, лінійних моделей (SVM, логістична регресія), ансамблевих алгоритмів, інструменти для крос-валідації тощо. На її основі легко будувати відтворювані експериментальні пайплайни, комбінуючи різні перетворення та моделі в єдину схему. Поряд з цим з’являються і спеціалізовані бібліотеки, такі як faststylometry, що пропонує швидку реалізацію методу дельти Берроуза [15], [16].

Для роботи з ембеддингами поширені бібліотеки sentence-transformers (обгортка над HuggingFace Transformers), які дозволяють легко використовувати попередньо навчені моделі MiniLM та інші трансформери. Вони забезпечують високорівневі інтерфейси для отримання векторних подань тексту, обчислення семантичної схожості та побудови класифікаторів поверх ембеддингів. Такий інструментарій дає змогу розробникам зосередитися на дизайні задачі та ансамблю, а не на низькорівневій реалізації алгоритмів.

Огляд цих рішень показує, що, попри наявність потужних інструментів, більшість з них або орієнтовані на англомовні/великі тексти,

або не враховують специфіку коротких україномовних повідомлень. Крім того, готові системи рідко надають зручний інтерфейс для поєднання класичних стилометричних ознак та нейронних ембеддингів у єдиному ансамблі в режимі профільної верифікації. Це обґрунтовує доцільність створення окремого інструментарію, адаптованого під наш предметний домен.

1.6 Вибір методів, інструментів та мов програмування

Враховуючи результати аналізу предметної галузі та специфіку коротких україномовних текстів, для реалізації системи авторської атрибуції обрано ансамблевий підхід, що поєднує кілька різнорідних моделей, але при цьому залишається відносно ресурсно ощадним і пояснюваним.

Як основну мову реалізації обрано Python, що забезпечує доступ до розвиненої екосистеми бібліотек для машинного навчання, обробки текстів і побудови звітів. Це дозволяє реалізувати повний цикл обробки – від попередньої підготовки даних до формування інтерпретованих результатів – у межах єдиного технологічного стеку.

Класичний канал ансамблю базується на:

- TF-IDF-представленні символьних n-грам (переважно довжини 3 – 5) з параметрами, налаштованими під короткі тексти (нормалізація, sublinear_tf, обмеження за мінімальною частотою);
- наборі ручних стилометричних ознак (частоти пунктуації, емодзі, великої літери, функційних слів, середні довжини слів і речень тощо), які формують writeprints-подібний густий вектор.

На цих ознаках навчається лінійна модель – SVM з подальшим калібруванням або логістична регресія, реалізована засобами scikit-learn. Вона добре працює з високорозмірними розрідженими векторами та є достатньо прозорою.

Другий компонент ансамблю – метод дельти Берроуза, реалізований через бібліотеку faststylometry. Він обчислює відстань між частотними профілями функційних слів автора й цільового тексту. Цей канал мало залежить від конкретної реалізації TF-IDF та є корисною «класичною» точкою відліку.

Нейромережевий компонент представлено семантичними ембеддингами, що обчислюються через sentence-transformers на базі легкої моделі класу MiniLM. Такий вибір дозволяє отримати стійке до перефразувань подання тексту й оцінювати косинусну подібність між ембеддингами без залучення надто важких моделей, придатних лише для GPU-середовищ.

Усі отримані сигнали – косинусна подібність у просторі TF-IDF, Delta-оцінки, косинусна подібність густих стилометричних векторів, косинус ембеддингів, а також додаткові дивергенції й різниці ручних ознак – інтегруються в єдиний вектор ознак для мета-класифікатора. На цьому рівні використовується лінійна логістична регресія, яка виконує роль ансамблевого калібратора і повертає ймовірність $p(\text{same\_author})$.

Для зберігання проміжних моделей і профілів застосовується joblib, що дозволяє кешувати векторизатори, класифікатори, калібратори та профілі авторів. Система доповнюється консольним інтерфейсом у вигляді окремого CLI-застосунку, який надає базові команди (тренування, побудова профілів, верифікація, порівняння двох текстів), а також демо-інтерфейсом на основі Streamlit для інтерактивного використання.

Окремим завданням є формування пояснюваних звітів щодо результатів верифікації авторства. Для цього реалізовано генерацію статичних HTML-звітів із вбудованими таблицями топових n-грам, стилометричних характеристик, Delta-оцінок та текстових пояснень. Такий звіт може бути використаний як додаток до технічного висновку в контексті інформаційної безпеки чи аналітичних досліджень.

1.7 Висновки до розділу 1

У цьому розділі було сформульовано задачу авторської атрибуції для коротких україномовних текстів та показано, чим вона відрізняється від класичних сценаріїв аналізу великих текстів. Проаналізовано ключові виклики, пов’язані з малим обсягом даних, неформальністю мовлення, код-світчингом, а також можливістю свідомого маскування стилю.

Розглянуто основні групи стилометричних ознак: лексичні, символьні, синтаксичні та ідіосинкратичні. Показано, що для коротких текстів особливо корисними є символьні n-грами, розподіли функційних слів, пунктуації, емодзі та інші низькорівневі патерни письма, які складно контролювати вручну. На цій основі обґрунтовано використання writeprints-подібного профілю стилю.

Проаналізовано основні класи методів авторської атрибуції – від статистичних відстаневих підходів (зокрема дельти Берроуза) та методів класичного машинного навчання на основі TF-IDF до нейромережевих моделей і трансформерних ембеддингів. Окрему увагу приділено специфіці задачі верифікації авторства та практиці ансамблевого комбінування, де кілька незалежних сигналів поєднуються мета-класифікатором.

Огляд існуючих інструментів показав, що доступні рішення або орієнтовані на інші мови та типи текстів, або не враховують специфіку коротких україномовних повідомлень. Це обґрунтувало вибір Python-орієнтованого технологічного стеку й ансамблю, до складу якого входять TF-IDF символьних n-грам з ручними стилометричними ознаками, метод дельти Берроуза та трансформерні ембеддинги, об’єднані логістичним мета-калібратором.

Таким чином, у розділі сформовано теоретичне та методологічне підґрунтя для подальшого проєктування архітектури системи авторської атрибуції, що буде розглянуто в наступному розділі.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі та ключові виклики

На основі проведеного аналізу предметної галузі та існуючих підходів до авторської атрибуції в цьому підрозділі формулюються функціональні та нефункціональні вимоги до розроблюваної системи. Вимоги задають рамки подальшого проєктування: визначають, які задачі система повинна підтримувати, у якому режимі працювати, які ресурси споживати та які якісні характеристики демонструвати.

З функціональної точки зору система повинна забезпечувати повний цикл роботи з моделями авторської атрибуції. Це означає, що користувач має мати змогу завантажити корпус текстів із мітками авторів, виконати його попередню обробку, навчити базові моделі, побудувати ансамблевий мета-класифікатор, а також зберегти всі отримані моделі й профілі у вигляді файлів для подальшого використання без повторного навчання. Важливим є й те, що система має працювати не лише в лабораторному режимі, а й у прикладному: користувач повинен мати можливість застосувати вже навчені моделі до нових текстів, не повертаючись кожного разу до етапу тренування.

Хоча основною задачею роботи є верифікація авторства, система не повинна обмежуватися лише цим режимом. У типових сценаріях дослідник або практичний користувач може захотіти як оцінити, чи належить певний текст конкретному автору, так і отримати розподіл ймовірностей по кількох авторам з набору. Тому система має підтримувати два споріднені режими: класифікацію авторства відносно фіксованого набору авторів та верифікацію авторства у форматі «профіль автора – цільовий текст» або «два тексти». У першому випадку результатом є, наприклад, найбільш ймовірний автор і відповідні числові оцінки каналів ансамблю; у другому

система повинна повертати ймовірність того, що тексти належать одній особі, а також бінарне рішення за попередньо налаштованим порогом.

Ключовим елементом функціональності є робота з профілями авторів. Система повинна дозволяти будувати узагальнений стилометричний профіль на основі набору текстів одного автора, зберігати його у вигляді файла, надалі завантажувати та використовувати для порівняння з новими текстами. У найпростішому випадку такий профіль може включати середній вектор TF-IDF символьних n-грам, усереднений вектор ручних стилометричних ознак, а також частотні розподіли функційних слів, пунктуації, емодзі та інших елементів, які описують «почерк» автора. Для зручності аналізу система повинна також підтримувати режим прямого порівняння двох текстів без попереднього профілювання, коли користувач просто передає пару текстів і отримує оцінку схожості та ймовірність спільного авторства.

З огляду на архітектуру ансамблю до системи висувається вимога підтримувати декілька незалежних каналів оцінки схожості. Для кожної пари «профіль / текст» або «текст / текст» необхідно вміти обчислювати, принаймні, косинусну подібність у просторі TF-IDF символьних n-грам, значення дельти Берроуза, косинусну подібність між густими стилометричними векторами, а за наявності відповідної моделі – і косинусну подібність у просторі трансформерних ембеддингів. Усі ці сигнали мають бути зібрані в єдиний вектор ознак, який подається на вхід мета-класифікатора, що повертає підсумкову ймовірність спільного авторства. Важливо, щоб окремі канали могли працювати незалежно: якщо, наприклад, ембеддинги недоступні через відсутність моделі або обмеження ресурсів, система повинна коректно працювати на основі доступних компонентів, а не завершуватися з помилкою.

Окремий блок вимог стосується засобів взаємодії з користувачем. Система має надавати зручний програмний інтерфейс, орієнтований насамперед на дослідника або інженера з даних. Базові операції – навчання

моделей, побудова профілів, верифікація авторства, порівняння текстів, генерація звітів – повинні виконуватися через командний інтерфейс із чітко задокументованими параметрами. На основі тих самих внутрішніх модулів може бути реалізований і простий інтерактивний веб-інтерфейс, але його наявність не є критичною для виконання завдань роботи. Значна увага приділяється й пояснюваності: система повинна вміти формувати зрозумілі звіти, у яких поряд із підсумковою ймовірністю відображаються проміжні показники каналів, характерні n-грами, ключові стилометричні відмінності між текстами, розподіли функційних слів тощо.

Вимоги до обробки текстів безпосередньо випливають зі специфіки домену. Система повинна коректно працювати з короткими україномовними текстами, що містять елементи, характерні для інтернет-комунікації, зокрема емодзі, хештеги, змішану мову та латиницю, навмисні орфографічні спотворення, надмірну пунктуацію. У межах прийнятої архітектури вона має зберігати максимально можливу кількість стилометричної інформації, пов’язаної з такими елементами, і не руйнувати їх повністю на етапах попередньої обробки.

Нефункціональні вимоги стосуються насамперед продуктивності, ресурсної ефективності, точності та зручності супроводу. Система має бути придатною для запуску на звичайному настільному комп’ютері або ноутбуці без обов’язкової наявності графічного прискорювача. Час обробки однієї пари текстів у режимі верифікації повинен залишатися прийнятним для інтерактивного використання, а використання оперативної пам’яті – помірним навіть при роботі з корпусами, що містять сотні авторів і тисячі коротких повідомлень. При цьому в процесі експериментальних досліджень, описаних у наступних розділах, буде сформульовано цільові показники точності (наприклад, за метриками accuracy, F1, AUC) і перевірено, чи задовольняє розроблена система цим вимогам.

Особливе значення мають надійність і відмовостійкість. Система повинна не лише видавати коректний результат на «ідеальних» даних, а й

адекватно реагувати на некоректні або неповні вхідні дані: порожні тексти, проблеми з кодуванням, відсутність міток автора, помилки під час завантаження моделей. У таких випадках очікується, що компоненти будуть повертати зрозумілі діагностичні повідомлення і, по можливості, продовжувати роботу без пошкодження наявних моделей та профілів. Архітектура має дозволяти масштабування: збільшення обсягу корпусу, кількості авторів чи кількості каналів ансамблю не повинно вимагати радикальної переробки системи, обмежуючись змінами конфігурації та апаратних ресурсів.

Ще одна група вимог стосується портативності та зручності супроводу. Система має працювати на поширених операційних системах (передусім Linux та Windows) за умови встановлення необхідних залежностей. Вимоги до версії інтерпретатора, бібліотек машинного навчання та інструментів для ембеддингів повинні бути чітко задокументовані. Структура проєкту, взаємодія основних модулів і формати збереження моделей мають бути описані так, щоб інший розробник міг без надмірних зусиль розширювати й підтримувати систему, додаючи нові типи ознак або нові канали ансамблю.

Окремо варто підкреслити вимоги щодо конфіденційності та безпеки даних. Оскільки система потенційно може застосовуватися до чутливих текстів (службова переписка, внутрішні документи, особисті повідомлення), вона повинна працювати повністю локально і не передавати дані до сторонніх сервісів. Усі операції з текстами й профілями виконуються в межах середовища користувача, а результати зберігаються в локальному файловому просторі.

Сукупність описаних функціональних і нефункціональних вимог задає орієнтири для подальшого проєктування архітектури системи авторської атрибуції. У наступних підрозділах на основі цих вимог буде побудована структура модулів, визначено основні компоненти ансамблю та описано принципи їхньої взаємодії.

2.2 Архітектура системи

Відповідно до сформульованих вимог архітектура системи авторської атрибуції побудована як багаторівнева, з чітким розділенням відповідальностей між підсистемами. Це дозволяє незалежно розвивати окремі компоненти (векторизатор, Delta, ембеддинги, мета-калібратор), не порушуючи загальної логіки роботи, а також спрощує подальшу експлуатацію та супровід.

У центрі рішення знаходиться Python-пакет authorship_attrib, всередині якого розташований модуль aa з основною логікою. Саме він містить засоби векторизації текстів, обчислення стилометричних ознак, побудови Delta-корпусу, калібрування базових методів, формування авторських профілів, виконання верифікації та побудови пояснюваних звітів. Навколо нього розташовані допоміжні компоненти: модулі попередньої обробки, утиліти для статистик, модуль командного інтерфейсу та, за потреби, інтерактивний демо-застосунок.

Архітектурно весь процес можна умовно розділити на кілька логічних рівнів. Перший рівень відповідає за попередню обробку текстів. У відповідному модулі реалізовані прості, але стійкі до шуму процедури: маскування URL-адрес, електронної пошти та згадок користувачів без втрати кінцевої пунктуації, а також розбиття тексту на речення за допомогою регулярних виразів. Додаткові утиліти відповідають за підрахунок емодзі, пунктуації та інших простих статистик, які надалі використовуються як частина стилометричних ознак. Завдяки цьому навіть дуже неформальні й фрагментарні повідомлення зберігають усі важливі для авторської атрибуції маркери, але при цьому не містять зайвих технічних артефактів. Такий підхід дозволяє надалі застосовувати одні й ті самі моделі до різних джерел текстів без істотної зміни логіки попередньої обробки. Цей рівень не займається жодним моделюванням, а лише готує текст у такому вигляді, щоб наступні модулі могли отримати максимально корисну

стилометричну інформацію. Архітектура ядра пакета authorship_attrib наведена на рисунку 2.1.

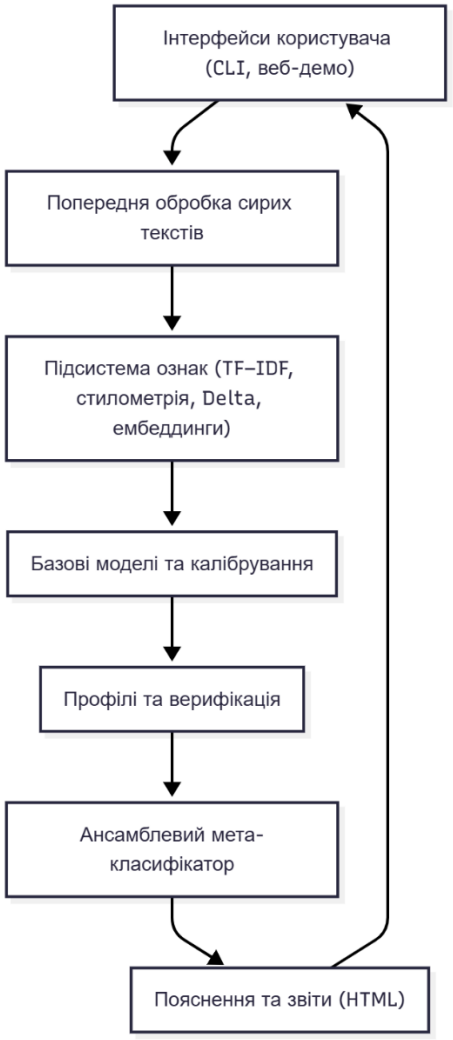

Рисунок 2.1 – Загальна архітектура системи авторської атрибуції

Другий рівень становить підсистема ознак. Вона реалізована у вигляді класу векторизатора, який поєднує кілька каналів ознак. Основу становить символьний TF-IDF з діапазоном n-грам, підібраним під короткі тексти, з відключеною примусовою знижкою регістру, сублінійною нормалізацією термів та відсіканням занадто рідкісних n-грам. Паралельно обчислюється набір густих ручних стилометричних характеристик: частоти пунктуації, емодзі, пропорція великих літер, середня довжина слів і речень, частоти

функційних слів згідно зі спеціальним словником. Окремий підканал цієї підсистеми відповідає за побудову Delta-корпусу: тексти токенізуються, нормуються, з них вилучаються функційні слова, після чого формується сегментований корпус у форматі, придатному для використання faststylometry [15]. На цьому ж рівні реалізовано ліниве підключення ембеддингового каналу: якщо в середовищі присутня модель MiniLM, векторизатор може за запитом обчислювати семантичні ембеддинги для текстів; у разі відсутності моделі фіксується попередження і ембеддинговий канал ігнорується.

Третій рівень складається з базових моделей і механізмів калібрування. Для класичного каналу використовується окремий класифікатор, який обгортає лінійну SVM-модель і засоби калібрування ймовірностей через CalibratedClassifierCV. Це дозволяє перетворити детерміновані рішення SVM на згладжену ймовірнісну шкалу, сумісну з мета-класифікатором. Паралельно для Delta-методу передбачено власний калібрувальний пайплайн, який на основі прикладів пар текстів «один автор» і «різні автори» навчає просту логістичну регресію, що перетворює сирі значення Delta на апроксимовану ймовірність спільного авторства [9], [10]. У результаті і класичний TF-IDF-канал, і Delta-підсистема, і ембеддинги (за наявності) подають на вихід не лише відстані або косинусні міри, а й узгоджені ймовірнісні оцінки.

Четвертий рівень – це підсистема профілів і верифікації, реалізована в окремому модулі. Вона відповідає за роботу у двох базових сценаріях: побудова профілю автора та перевірка нового тексту. Під час побудови профілю модуль приймає набір текстів одного автора, пропускає їх через векторизатор, агрегує отримані ознаки (наприклад, усереднює TF-IDF-вектори, обчислює середні стилометричні метрики, формує Delta-корпус) і зберігає отриманий опис разом із посиланнями на відповідні моделі. Під час верифікації цільовий текст проходить той самий шлях, після чого модуль обчислює набір показників подібності між профілем і новим текстом для

всіх доступних каналів: косинусну схожість у просторі TF-IDF, ймовірнісну оцінку Delta, косинусну схожість стилометричних векторів, косинус ембеддингів, дивергенції розподілів функційних слів тощо. Ці показники формують вектор ознак для мета-класифікатора.

На цьому ж рівні реалізований сценарій прямого порівняння двох текстів без явного профілю. В цьому випадку кожен текст тимчасово розглядається як профіль, для обох обчислюються ознаки, після чого формується симетричний набір показників подібності. Це дозволяє використовувати однакову логіку як для верифікації «профіль – текст», так і для швидкого порівняння «текст – текст».

П’ятий рівень становить ансамблевий мета-класифікатор. Він не «знає» про внутрішні деталі TF-IDF, Delta чи ембеддингів, а працює лише з числовим вектором, який містить значення косинусних подібностей, ймовірностей, дивергенцій і різниць ручних стилометричних метрик. На цьому векторі навчається лінійна логістична регресія, яка виконує роль підсумкового калібратора: на основі набору сигналів від різних каналів вона повертає оцінку ймовірності того, що тексти належать одному автору. Такий поділ дозволяє легко замінювати або додавати канали: достатньо розширити вектор ознак і перевчити мета-класифікатор, не змінюючи загальну архітектуру.

Важливим елементом архітектури є підсистема пояснень і звітності. Окремі модулі відповідають за ранжування символьних n-грам за внеском у рішення, обчислення різниць у writeprints-подібних ознаках, оцінку розподілів функційних слів, пунктуації, емодзі. На основі результатів верифікації та цих допоміжних метрик формується статичний HTML-звіт, який містить шкалу ймовірності, таблиці найбільш показових n-грам, порівняльні графіки стилометричних характеристик і список попереджень (наприклад, про надто короткий текст або відсутність калібрування Delta). Архітектурно ця підсистема побудована так, щоб не

впливати на сам процес прийняття рішення: вона лише читає вже обчислені ознаки й результати, не змінюючи їх.

Нарешті, на верхньому рівні знаходяться засоби взаємодії з користувачем. Командний інтерфейс реалізований як окремий модуль, що підключає основний пакет і надає набір команд для тренування, побудови профілів, верифікації та генерації звітів. Паралельно існує простий інтерактивний застосунок на базі веб-фреймворку, який демонструє типові сценарії використання, але не є обов’язковою частиною для розгортання ядра системи. Усі ці зовнішні оболонки звертаються до одних і тих самих внутрішніх API, що дозволяє відокремити логіку обробки та моделювання від конкретної форми представлення користувачеві.

У такому вигляді архітектура системи забезпечує одночасно модульність, розширюваність і можливість пояснення рішень. Наступні підрозділи будуть присвячені детальнішому опису реалізації ключових компонентів цієї архітектури: векторизатора, базових моделей, мета-класифікатора та засобів побудови профілів і звітів.

2.3 Опис основних модулів системи

Описана в попередньому підрозділі архітектура реалізується через набір узгоджених між собою модулів, кожен з яких відповідає за окремий аспект роботи системи авторської атрибуції. Таке розділення дозволяє ізолювати обчислювально важкі частини (наприклад, векторизацію або обчислення Delta), логіку побудови профілів, процедури калібрування та зовнішні інтерфейси. Подібна структура не лише спрощує супровід і тестування окремих компонентів, а й полегшує подальше доповнення системи новими каналами ознак або альтернативними моделями. За потреби окремі модулі можуть бути замінені або розширені без зміни загальної схеми взаємодії між частинами системи. У цьому підрозділі розглядаються

основні модулі ядра системи, їхня роль та взаємодія між собою, їх схема наведена на рисунку 2.2.

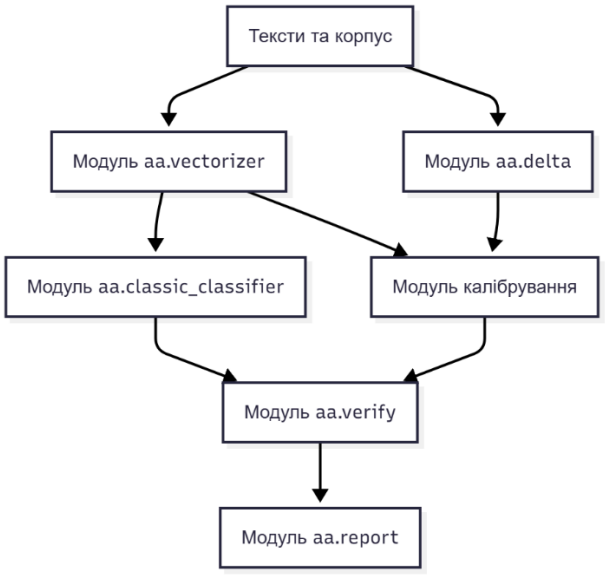

Рисунок 2.2 – Структура основних модулів ядра системи

Центральну роль відіграє модуль aa.vectorizer, у якому зосереджено логіку побудови ознак. В межах цього модуля реалізовано навчання та застосування векторизатора, що поєднує символьний TF-IDF-канал із набором ручних стилометричних ознак. Під час тренування векторизатор аналізує вхідний корпус, обирає діапазон n-грам, формує словник, обчислює статистики для TF-IDF та налаштовує параметри відсікання рідкісних ознак [16]. Одночасно закладається схема обчислення густого стилометричного вектора: визначається перелік функційних слів, типи пунктуації та емодзі, показники довжини слів і речень, які згодом будуть перетворюватися на числові значення. Той самий векторизатор у режимі застосування приймає нові тексти, виконує необхідну попередню обробку, віддає TF-IDF-представлення та стилометричні ознаки й, за наявності відповідної моделі, обчислює семантичні ембединги. Всі ці дані зберігаються у внутрішньо узгоджених структурах, що дозволяє надалі

передавати їх у модулі класифікації та верифікації без ручного дублювання логіки.

Модуль aa.delta відповідає за роботу з методом дельти Берроуза. У ньому реалізовано підготовку спеціального корпусу для Delta: тексти приводяться до потрібного формату, сегментуються на фрагменти, виділяються функційні слова, будуються частотні профілі авторів та цільових текстів. На основі цих профілів модуль обчислює сирі значення Delta-збіжності, тобто середні модульні відхилення, а також надає обгортку, яка, отримавши вже навчені параметри, може швидко повертати Delta-оцінки для нових текстів. Тут же передбачений інтерфейс до калібрувальної логіки: сирі значення Delta можуть передаватися в окрему модель логістичної регресії, яка перетворює їх на узгоджену ймовірнісну шкалу [15].

Для класичного каналу класифікації використовується модуль aa.classic_classifier. Він інкапсулює в собі навчання лінійної моделі на TF-IDF-ознаках, зокрема створення та налаштування лінійної SVM або логістичної регресії, а також застосування процедури калібрування ймовірностей. Під час тренування модуль отримує векторизатор та розмічений корпус, навчає базовий класифікатор і обгортає його в схему, що дозволяє для кожного тексту або пари текстів отримати не лише класове рішення, а й оцінку ймовірності. У режимі застосування цей модуль приймає підготовлені векторизатором представлення текстів, повертає оцінки для кожної гіпотези авторства або, у разі парної верифікації, для гіпотези «один автор / різні автори».

Окремим важливим компонентом є модуль калібрування, який реалізує допоміжний клас калібратора косинусної подібності. Саме він відповідає за перетворення різних сирих метрик схожості на спільну ймовірнісну шкалу. Для цього формується навчальний набір пар текстів із мітками «той самий автор» і «різні автори», для кожної пари обчислюється набір числових характеристик (косинусні схожості, Delta, дивергенції

розподілів, різниці ручних ознак), після чого над цими ознаками навчається проста логістична регресія. Отриманий калібратор зберігає інформацію про версію формату, перелік ознак, статистики й може бути завантажений разом із профілем автора. У випадках, коли навчальний датасет для калібрування відсутній або недостатній, модуль передбачає запасну поведінку: застосування гладкої сигмоїдальної функції до косинусної подібності як наближення ймовірності, що дозволяє системі працювати навіть без повноцінного калібрування, хоч і з меншою точністю.

Модуль aa.verify реалізує високорівневі сценарії побудови профілів і верифікації. Його внутрішня логіка побудована навколо кількох функцій. По-перше, передбачено створення тимчасового профілю на основі одного тексту, який використовується в режимі швидкого порівняння двох текстів: модуль формує профіль для кожного тексту «на льоту», обчислює необхідні ознаки та попереджає користувача, якщо довжина текстів є надто малою для надійних стилометричних висновків. По-друге, реалізовано процедуру побудови стабільного профілю автора, яка очікує набір текстів достатнього обсягу, агрегує TF-IDF-представлення, стилометричні ознаки, ембедингиа (за наявності), формує Delta-корпус, а також вбудовує до профілю параметри калібратора. По-третє, модуль містить власне верифікацію: він приймає профіль і цільовий текст, звертається до векторизатора, Delta-модуля та ембеддингів, збирає усі доступні сигнали схожості, формує вектор ознак для мета-класифікатора і повертає підсумкову ймовірність спільного авторства разом із набором проміжних значень, які можуть бути використані в звіті.

Створення пояснюваних звітів зосереджено в модулі aa.report. Він не впливає на саме рішення, а працює як надбудова над результатами верифікації. Структура пояснювального звіту зображена на рисунку 2.3. Цей модуль отримує вхідний профіль, цільовий текст, набір обчислених ознак та підсумкову оцінку ймовірності, після чого формує структурований HTML-документ. До такого звіту зазвичай входять короткий вербальний опис

рішення (зокрема попередження про обмеження на кшталт малої довжини тексту), шкала ймовірності, таблиці з найхарактернішими символьними n-грамами для кожної сторони, графіки розподілу функційних слів і прості діаграми для ручних стилометричних характеристик.

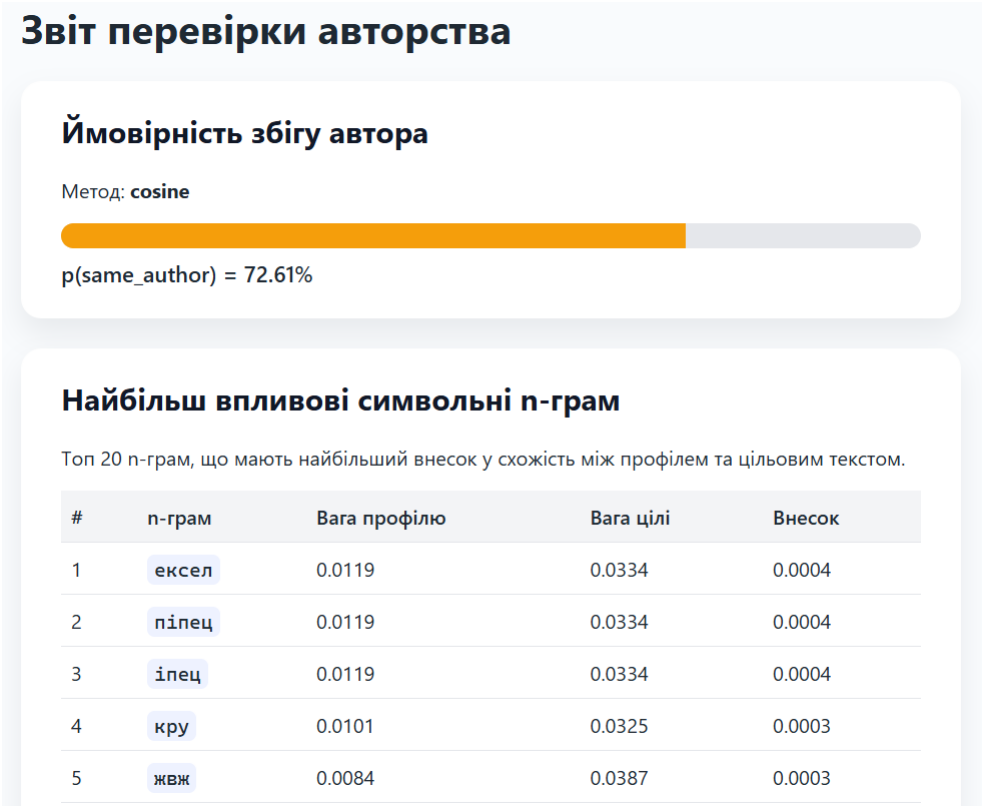

Рисунок 2.3 – Структурований пояснювальний звіт

Над ядром системи розташовується модуль командного інтерфейсу, який пов’язує описані вище компоненти в завершені сценарії використання. Він реалізує декілька базових команд: навчання моделей на каталозі текстів, побудову профілю автора на основі директорії або стовпця CSV-файла, верифікацію тексту проти існуючого профілю з можливістю збереження HTML-звіту, а також швидке порівняння двох текстів. Кожна команда відповідає за розбір параметрів командного рядка, перевірку коректності шляхів і конфігурації, а потім передає роботу у відповідні функції модулів. Завдяки цьому користувач взаємодіє з цілісними процедурами на кшталт

«навчити модель на заданому корпусі» або «перевірити текст проти профілю автора», що зменшує ризик помилок конфігурації. У разі потреби ті самі команди можуть запускатися з зовнішніх скриптів або інтегруватися в інші робочі процеси, перетворюючи систему на придатний для автоматизації інструмент, а не лише на експериментальний прототип. На рисунку 2.4 проілюстровано роботу цього модуля.

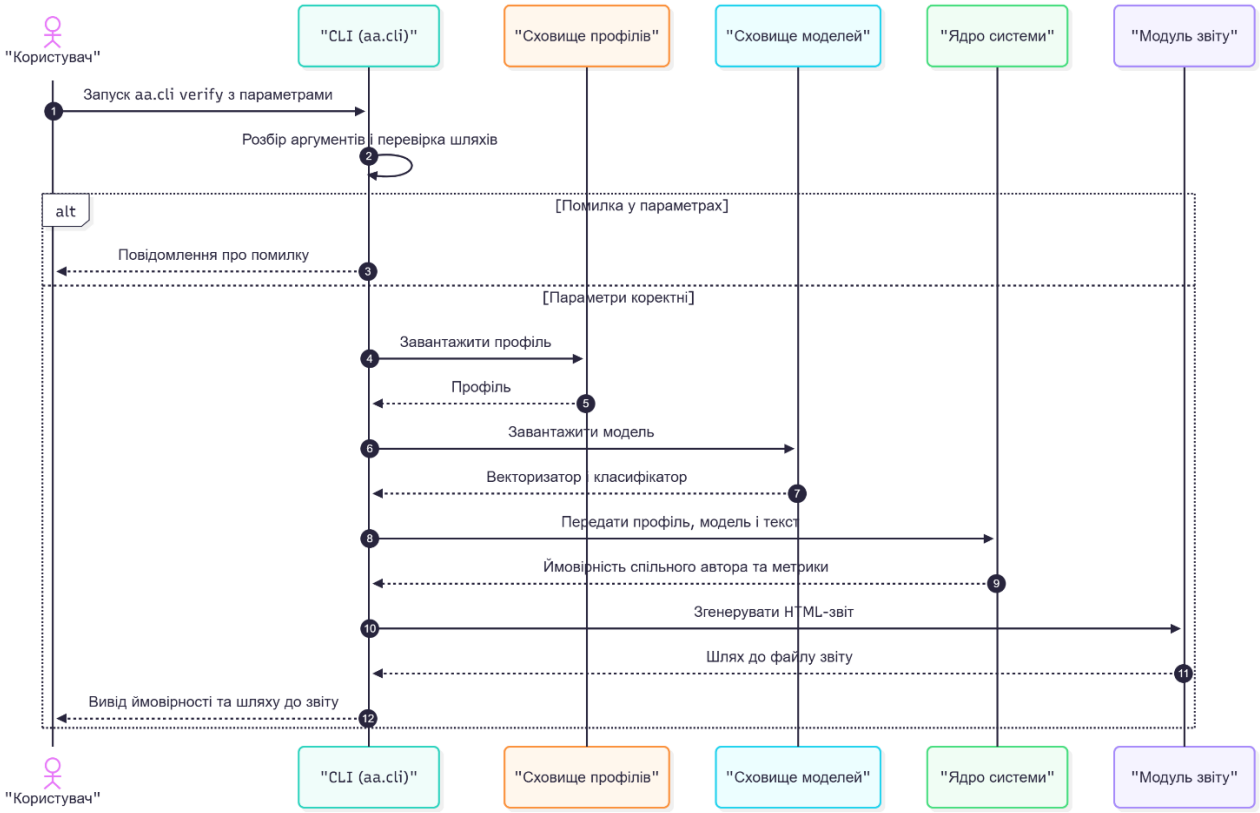

Рисунок 2.4 – UML-діаграма послідовності виконання команди верифікації

Додатковим, але не обов’язковим для базової функціональності елементом є інтерактивний демо-застосунок, реалізований у вигляді окремого файлу зі стартовою функцією веб-інтерфейсу. Він використовує той самий внутрішній API, що й командний інтерфейс, і дозволяє завантажувати тексти, обирати профілі, запускати верифікацію та переглядати згенеровані звіти в браузері. З точки зору архітектури цей модуль є лише альтернативною оболонкою над ядром системи.

Таким чином, набір модулів authorship_attrib відображає загальну архітектуру системи: попередня обробка та побудова ознак, базові моделі й калібрування, профілі та верифікація, пояснення та зовнішні інтерфейси. Це дозволяє у наступному розділі перейти до детального опису реалізації кожного компонента.

2.4 Вибір алгоритмів та моделей

У попередніх підрозділах було сформульовано вимоги до системи авторської атрибуції та описано її загальну архітектуру. На цьому етапі необхідно обґрунтувати вибір конкретних алгоритмів та моделей, які реалізують окремі компоненти ансамблю. Вибір робиться з урахуванням кількох ключових чинників: специфіки коротких україномовних текстів, обмежених обчислювальних ресурсів, необхідності пояснюваності результатів та потреби отримати стійкий до шуму інструмент верифікації авторства. На рисунку 2.5 показано чотири канали: TF-IDF символьних n-грам, Delta, густі стилометричні ознаки, ембеддинги MiniLM, а також мета-класифікатор, що об’єднує їхні виходи.

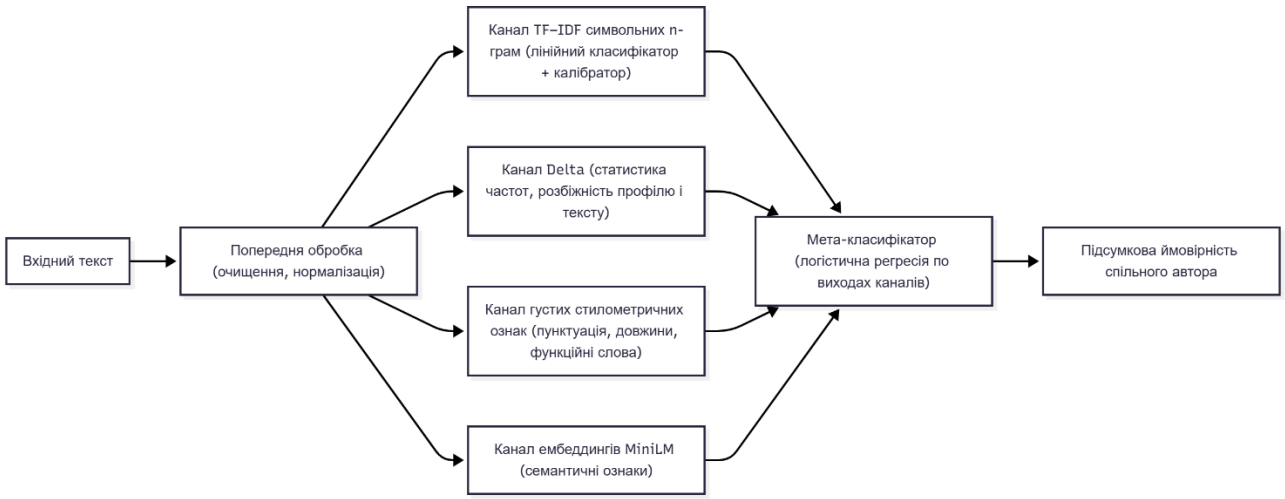

Рисунок 2.5 – Ансамбль каналів ознак та мета-класифікатора

Базовим елементом ансамблю обрано класичний канал на основі TF-IDF символьних n-грам. Як показує практика стилометричних досліджень, символьні n-грами краще за словні працюють на коротких і неформальних текстах, де граматика часто порушується, а слова можуть бути навмисно спотворені. Додатковою перевагою є мовна та орфографічна незалежність: той самий механізм однаково добре реагує на кирилицю, латиницю, емодзі та змішаний текст. У цьому каналі для класифікації обрано лінійну модель з сімейства SVM або логістичної регресії: такі моделі добре поводяться в просторі високорозмірних розріджених ознак, є стійкими до надлишковості ознак та відносно просто інтерпретуються через вагові коефіцієнти. Додатково поверх базового класифікатора використовується калібрування ймовірностей, що дозволяє працювати не тільки з «жорсткими» рішеннями, а й із плавною ймовірнісною оцінкою, сумісною з ансамблевим мета-класифікатором.

Другим складником ансамблю є канал на основі методу дельти Берроуза. Цей підхід, розглянутий у підрозділі 1.4.1, показав свою ефективність у задачах авторської атрибуції для різних мов, зокрема завдяки чутливості до розподілу функційних слів та стійкості до зміни тематики. Для коротких текстів Delta не завжди дає достатньо стабільні оцінки, проте в поєднанні з іншими каналами її показник є корисним незалежним сигналом. У запропонованій системі Delta використовується не як окремий кінцевий класифікатор, а як джерело числової оцінки подібності, яка надалі калібрується за допомогою простої логістичної моделі. Таким чином, сире значення Delta, перетворюється на ймовірнісну оцінку, яку можна безпосередньо порівнювати з іншими каналами.

Третій канал відповідає за густі стилометричні ознаки, близькі за духом до концепції Writeprints. На відміну від TF-IDF, який працює з великою кількістю локальних символьних патернів, цей канал фокусується на агрегованих характеристиках стилю: частоті різних типів пунктуації, пропорції великих літер, розподілі довжин слів і речень, частотах

функційних слів, кількості емодзі тощо. Сукупність таких ознак формує компактний вектор, який описує текст або профіль автора в просторі «глобальних» стилістичних параметрів. Для порівняння тексту та профілю в цьому просторі використовується косинусна подібність, а також додаткові статистичні міри на кшталт χ^2 та дивергенції між розподілами функційних слів. На основі цих величин формується підканал ознак, який, з одного боку, є відносно маловимірним і интерпретованим, а з іншого – доповнює TF-IDF, що фіксує більш локальні патерни.

Четвертий канал пов’язаний із нейромережевими моделями на основі семантичних ембеддингів. Як компроміс між якістю та ресурсною ефективністю обрано використання попередньо навченої моделі класу MiniLM із бібліотеки sentence-transformers. Такі моделі дозволяють перетворювати текст у щільний вектор невеликої розмірності, який містить інформацію про семантичний вміст і частково відображає стиль. Важливою перевагою є можливість переносити знання з великих зовнішніх корпусів, що частково компенсує обмежений обсяг навчальних даних у рамках конкретного проєкту. Для кожної пари «профіль – текст» або «текст – текст» у цьому каналі обчислюється косинусна подібність ембеддингів. Особливість проєктування полягає в тому, що ембеддинговий канал вважається опційним: якщо модель недоступна або робота здійснюється в суворо обмеженому середовищі, ансамбль залишається працездатним, хоч і з дещо гіршими показниками якості.

Усі перелічені канали працюють як окремі «експерти», що генерують числові оцінки схожості між текстом і профілем автора або між двома текстами. Для об’єднання цих оцінок обрано лінійну логістичну регресію, яка виконує роль мета-класифікатора. Таке рішення має кілька переваг. По-перше, логістична регресія добре підходить для задачі бінарної класифікації з невеликою кількістю вхідних ознак (у нашому випадку – кілька десятків числових характеристик, що описують різні канали). По-друге, її вагові коефіцієнти легко інтерпретувати: за ними можна оцінити відносний внесок

кожного каналу в підсумкове рішення, що важливо для пояснюваності системи. По-третє, лінійна модель знижує ризики перенавчання в умовах обмеженого обсягу пар «один автор / різні автори», які доступні для тренування мета-рівня.

Навчання мета-класифікатора відбувається на спеціально сформованому наборі пар текстів. Для кожної пари обчислюється вектор ознак, що включає косинусні подібності в просторі TF-IDF, густих стилометричних векторів та ембеддингів, значення Delta (як сире, так і після попереднього локального калібрування), дивергенції між розподілами функційних слів, різниці в окремих стилометричних показниках, а також допоміжні характеристики, такі як довжина текстів. На основі цього набору мета-класифікатор навчається відрізняти пари «один автор» від пар «різні автори» і в результаті повертає оцінку ймовірності спільного авторства для нових пар.

У рамках проєктування також було розглянуто альтернативні варіанти, передусім глибокі нейромережеві архітектури з сіамськими або триплетними структурами, які безпосередньо навчають простір ембеддингів з урахуванням міток «той самий автор / різні автори». Проте такі підходи мають низку суттєвих недоліків у контексті даної роботи: вони вимагають значно більших обсягів навчальних даних, чутливі до дисбалансу класів і потребують потужніших обчислювальних ресурсів, бажано із використанням графічних прискорювачів. Крім того, їх пояснюваність значно нижча: важко відокремити вплив стилістичних та змістових факторів, а просте порівняння окремих каналів втрачається. З огляду на це перевагу було віддано більш «класичному» ансамблевому дизайну з чітко виділеними каналами і лінійним мета-класифікатором.

Окремою вимогою при виборі алгоритмів було забезпечення можливості їхнього використання в режимі офлайн на центральному процесорі. Саме тому всі компоненти ансамблю базуються на інструментах, доступних у звичайному Python-середовищі без обов’язкової GPU-

підтримки: символьний TF-IDF і лінійні моделі машинного навчання для класичного каналу, реалізація дельти Берроуза через спеціалізовану бібліотеку, легка трансформерна модель MiniLM для ембеддингів, логістична регресія для мета-рівня. Такий вибір дозволяє запускати систему як на робочій станції, так і на сервері без суттєвих змін у конфігурації.

Таким чином, у підрозділі було обґрунтовано вибір набору алгоритмів, з яких складається ансамбль: класичної моделі на TF-IDF символьних n-грам, методу дельти Берроуза, каналу густих стилометричних ознак та нейромережевого каналу на основі ембеддингів MiniLM, об’єднаних лінійною логістичною регресією на рівні мета-класифікатора. Така композиція дозволяє поєднати сильні сторони різних підходів і компенсувати їхні слабкі місця, забезпечуючи при цьому прийнятний баланс між точністю, ресурсними витратами та пояснюваністю рішень. У наступному розділі ці концептуальні вибори будуть конкретизовані на рівні реалізації та експериментальних налаштувань.

2.5 Проєктування форматів збереження моделей, профілів та звітів

Ефективність і зручність використання системи авторської атрибуції значною мірою залежать не лише від обраних алгоритмів, а й від того, як організовані формати збереження проміжних і кінцевих артефактів. Йдеться про навчальні моделі, стилометричні профілі авторів, параметри калібрування, результати верифікації та пояснювані звіти. У практичних умовах важливо, щоб ці артефакти могли бути легко перенесені між різними середовищами, мали однозначну структуру і не вимагали доступу до вихідного коду для базової інтерпретації. Окремого значення набуває розділення «стабільних» об’єктів (навчені моделі, профілі) та «одноразових» результатів експериментів (проміжні таблиці, журнали запусків), що полегшує супровід і прибирання застарілих даних. Невдало спроєктована схема збереження ускладнює відтворюваність експериментів,

перенесення системи між середовищами й подальший розвиток проєкту. На рисунку 2.6 зображено схему каталогу для моделей, профілів та звітів.

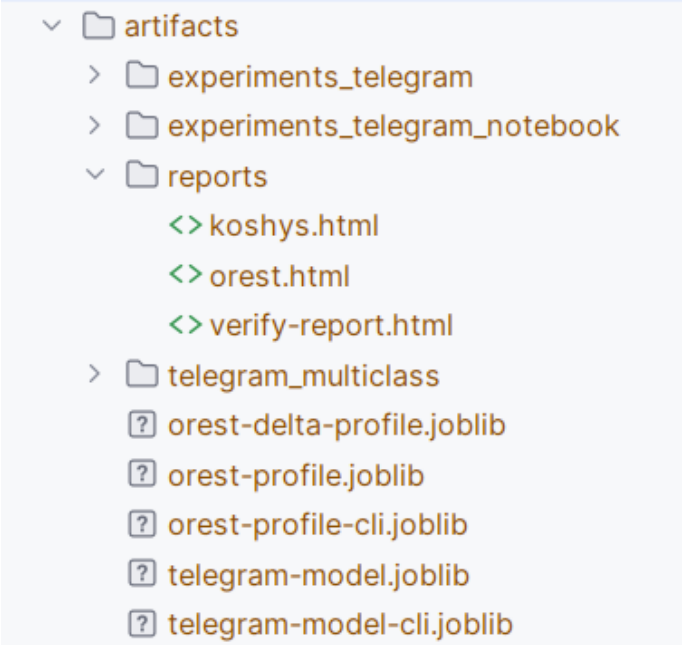

Рисунок 2.6 – Схема каталогу для моделей, профілів та звітів

Основною вимогою до форматів збереження є поєднання трьох властивостей: портативності, відтворюваності та практичності. Портативність означає, що артефакти можна переносити між різними інсталяціями системи без додаткової ручної обробки. Відтворюваність передбачає можливість відновити стан моделі, профілю або експерименту через певний час, навіть після оновлення бібліотек чи змін конфігурації. Практичність стосується того, наскільки легко ці артефакти інтегруються в робочий процес: чи можна їх швидко завантажити, додати до репозиторію, архівувати тощо.

Для збереження моделей машинного навчання, векторизаторів та калібраторів обрано бінарний формат на основі бібліотеки joblib [17]. Такий вибір зумовлений тим, що joblib оптимізовано саме під об’єкти з великою кількістю числових параметрів (матриці ваг, розріджені TF-IDF-вектори тощо) і забезпечує як стиснення, так і відносно швидке завантаження. Кожен основний компонент ансамблю (векторизатор, класичний

класифікатор, мета-класифікатор, модуль дельти з його статистиками) зберігається окремим файлом у спеціальному каталозі моделей. Такий поділ дозволяє незалежно оновлювати або замінювати окремі частини системи: наприклад, перевчити лише мета-класифікатор без зміни векторизатора або навпаки. Крім того, збереження кожної моделі в окремому файлі спрощує аналіз і налагодження, оскільки можна завантажити й протестувати конкретний компонент у відриві від решти системи.

Профілі авторів мають іншу природу: вони поєднують як числові структури (усереднені TF-IDF-вектори, стилометричні ознаки, ембедінги), так і описову інформацію (ідентифікатор автора, джерело даних, часові межі, інформацію про версію моделей). Тому для них використовується гібридний підхід. Числові частини профілю, які є громіздкими (наприклад, середній TF-IDF-вектор або агреговані ембедінги), зберігаються в бінарному форматі joblib, тоді як метадані фіксуються у вигляді окремого файла у форматі JSON. Такий поділ дозволяє, з одного боку, уникнути складності читання великих бінарних об’єктів людиною, а з іншого – зберігати всю службову інформацію в текстовій, легко переглядній формі. У метаданих профілю зберігаються, зокрема, версія схеми профілю, версія пакета, імена та контрольні суми файлів моделей, статистики щодо навчального корпусу (кількість текстів, середня довжина, дата побудови профілю), а також список активних каналів ансамблю на момент його створення.

Окрему увагу приділено збереженню параметрів навчання й конфігурації експериментів. Щоб забезпечити відтворюваність результатів, у каталозі кожного експерименту передбачається наявність текстового файла конфігурації у форматі YAML або JSON, де фіксуються: назва та опис експерименту, шлях до вхідного корпусу, перелік попередньо обраних авторів, параметри векторизації (діапазон n-грам, розмір словника, параметри нормалізації), налаштування Delta, список активно використаних стилометричних ознак, вибір трансформерної моделі для ембедінґів,

параметри розбиття на навчальну та тестову вибірки, значення випадкових зерен (seed). Завдяки цьому, маючи конфігураційний файл та відповідний корпус, користувач може в будь-який момент повторити навчання й переконатися в стабільності результатів або адаптувати налаштування під нові дані.

Результати верифікації та порівняння текстів зберігаються у двох формах. По-перше, у вигляді компактних машинно-читаних записів (наприклад, рядків CSV або записів JSON), де для кожної перевірки фіксуються ідентифікатор профілю, ідентифікатор тексту чи пари текстів, підсумкова ймовірність спільного авторства, бінарне рішення, а також окремі числові значення каналів ансамблю. Такі записи зручні для статистичного аналізу, побудови графіків залежності якості від довжини тексту, порівняння різних конфігурацій системи. По-друге, для вибраних випадків формуються пояснювані звіти у форматі HTML, які зберігаються в окремому каталозі звітів. Кожен звіт має унікальне ім’я, що включає дату та час генерації, а також короткий ідентифікатор перевірки, що дозволяє пов’язати його з відповідним записом у машинно-читаному журналі.

HTML обрано як основний формат для звітів з кількох причин. По-перше, він легко відчиняється у будь-якому сучасному браузері й дозволяє поєднати текстові пояснення, таблиці, прості графіки й кольорові маркування без потреби в додаткових інструментах. По-друге, статичний HTML можна передавати як окремий файл, архівувати, додавати до додатків дипломної роботи або внутрішніх звітів, не залежачи від середовища виконання системи. По-третє, HTML-шаблони дають змогу чітко відокремити логіку формування результатів від їхнього оформлення: модуль звітів отримує числові значення й текстові фрагменти і просто підставляє їх у наперед визначену структуру.

Важливою частиною проєктування форматів збереження є управління версіями. Оскільки система може розвиватися, змінювати представлення ознак і структуру профілів, у кожному бінарному та текстовому артефакті

зберігається явна інформація про версію формату й версію пакета, яким цей артефакт було створено. Це дозволяє, по-перше, відмовитися від використання застарілих моделей або профілів у новій версії системи без явного перетворення, а по-друге, за потреби реалізувати процедури міграції: наприклад, завантажити профіль старої версії, автоматично дообчислити нові стилометричні ознаки й зберегти його в оновленому форматі. Наявність таких маркерів версій також полегшує налагодження: якщо під час завантаження виникає помилка, система може явно повідомити, що цей файл був створений іншою, несумісною версією.

Окремо слід зазначити, що всі формати збереження орієнтовані на локальну роботу й не передбачають відправлення даних на зовнішні сервіси. Це узгоджується з нефункціональними вимогами щодо конфіденційності: тексти, профілі авторів, журнали верифікацій і звіти залишаються в межах файлової системи користувача. У випадку, якщо система застосовується до чутливих або конфіденційних даних, достатньо вжити стандартних заходів захисту файлового середовища, без додаткових мережевих налаштувань.

Таким чином, проєктування форматів збереження моделей, профілів та звітів забезпечує для системи авторської атрибуції необхідний баланс між ефективністю, відтворюваністю, переносимістю та безпекою. Чітка структура каталогів, поєднання бінарних форматів для числових артефактів і текстових форматів для метаданих та конфігурації утворюють основу для стабільної експлуатації й подальшого розвитку системи, а також спрощують проведення експериментів, результати яких будуть розглянуті в наступних розділах.

2.6 Висновки до розділу 2

У другому розділі було сформовано та послідовно обґрунтовано проєктні рішення, на яких базується система авторської атрибуції для коротких україномовних текстів соціальних мереж. Вихідною точкою стали

сформульовані функціональні та нефункціональні вимоги, які задали рамки подальшого проєктування. Зокрема, було визначено, що система має підтримувати як класифікацію авторства, так і верифікацію у форматі «профіль автора – текст» та «два тексти», працювати в умовах обмежених обчислювальних ресурсів, забезпечувати локальність обробки, конфіденційність даних та пояснюваність результатів.

На основі цих вимог була запропонована багаторівнева архітектура, у центрі якої знаходиться ядро пакета з модулем, що відповідає за побудову ознак, базові моделі, профілі та верифікацію, а зовні розташовані підсистеми попередньої обробки, звітності та зовнішні інтерфейси. Такий підхід дозволив чітко розділити відповідальності між компонентами: окремо виділено рівень попередньої обробки тексту, рівень побудови ознак (символьні TF-IDF, густі стилометричні характеристики, Delta-корпус, ембединги), рівень базових моделей і калібрування, рівень профілів та верифікації, а також рівень мета-класифікації й звітів.

Ця архітектурна схема була конкретизована через опис основних модулів системи. Було показано, як модуль векторизатора поєднує символьні TF-IDF і ручні стилометричні ознаки, як модуль Delta готує спеціалізований корпус і обчислює значення міри, як модуль класичного класифікатора інкапсулює навчання та калібрування лінійної моделі, як модуль верифікації реалізує сценарії побудови профілю та порівняння текстів, а модуль звітів формує пояснювані HTML-документи. Окремо наголошено, що зовнішні інтерфейси (командний рядок, демо-застосунок) є тонкими оболонками над цими модулями й не впливають на внутрішню логіку системи.

Подальший аналіз був присвячений вибору конкретних алгоритмів та моделей для кожного з каналів ансамблю. Було обґрунтовано використання символьного TF-IDF у поєднанні з лінійною моделлю (SVM або логістична регресія) як основи класичного каналу, застосування методу дельти Берроуза як відносно простого, але стійкого до тематичних змін індикатора

схожості стилю, формування writeprints-подібного вектора густих стилометричних ознак, а також використання попередньо навченої трансформерної моделі класу MiniLM для обчислення семантичних ембеддингів. Усі ці канали розглядаються як незалежні «експерти», чиї числові оцінки схожості об’єднуються лінійною логістичною регресією на рівні мета-класифікатора. Такий підхід дозволяє поєднати сильні сторони різних методів і зменшити ймовірність їхніх індивідуальних помилок, зберігаючи при цьому прийнятну пояснюваність рішення.

Важливою складовою проєктування стала організація форматів збереження моделей, профілів та звітів. Було показано, що використання бінарних форматів на основі joblib для числових артефактів (моделі, векторизатори, агреговані вектори профілів) у поєднанні з текстовими форматами для метаданих і конфігурацій (JSON або YAML) забезпечує необхідний баланс між ефективністю, портативністю та відтворюваністю. Окремо було обґрунтовано вибір HTML як формату для пояснюваних звітів, а також запроваджено систему явного маркування версій артефактів, що спрощує міграцію між різними версіями системи та запобігає використанню несумісних профілів і моделей.

Узагальнюючи, цей розділ задає цілісну концепцію системи авторської атрибуції: від вимог і загальної архітектури до конкретних алгоритмів і форматів збереження. В результаті спроєктовано модульну ансамблеву систему, орієнтовану на роботу з короткими україномовними текстами, здатну функціонувати в умовах обмежених ресурсів, забезпечувати локальну й конфіденційну обробку даних і надавати пояснювані результати верифікації авторства. У наступному розділі увага буде зосереджена на практичній реалізації спроєктованих компонентів, описі середовища розробки, структури даних та конкретних кроків побудови моделей і профілів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Середовище розробки та інструментальні засоби

Розробка системи авторської атрибуції для коротких україномовних текстів виконувалася з опорою на сучасні інструменти машинного навчання та обробки текстів, орієнтовані на мову програмування Python. Вибір середовища та інструментальних засобів визначався вимогами до відтворюваності експериментів, ресурсної ефективності, можливості локального запуску без графічного прискорювача, а також необхідністю підтримувати повний цикл роботи з моделями: від підготовки даних до формування пояснюваних звітів.

Основним засобом реалізації стала мова програмування Python версії 3.13. Вона забезпечує доступ до розвиненої екосистеми бібліотек для машинного навчання, роботи з текстом, наукових обчислень і побудови візуалізацій. Використання єдиної мови для всіх етапів обробки – від попередньої підготовки корпусу до генерації HTML-звітів – дозволяє уникнути зайвих накладних витрат на інтеграцію різнорідних технологій і спрощує супровід системи.

Базовий стек бібліотек включає декілька ключових компонентів. Для реалізації класичних алгоритмів машинного навчання та TF-IDF-векторизації використано бібліотеку scikit-learn, яка надає готові реалізації лінійних моделей (логістична регресія, лінійний SVM), інструменти для крос-валідації, налаштування гіперпараметрів і калібрування ймовірностей [16]. Обчислювальну основу становлять бібліотеки numpy та scipy, які відповідають за роботу з багатовимірними масивами та розрідженими матрицями, що є критично важливим при обробці високорозмірних TF-IDF-представлень [18], [19].

Для обчислення семантичних ембеддингів тексту застосовано бібліотеку sentence-transformers, яка надає зручний інтерфейс до попередньо

навчених трансформерних моделей класу MiniLM. Такий вибір дозволяє отримувати компактні, але інформативні векторні подання текстів без необхідності самостійного навчання великих нейромережевих моделей. Канал ембеддингів інтегрується в систему як опційний: у середовищах з обмеженими ресурсами його можна вимкнути, залишивши активними лише класичні методи [13].

Для реалізації методу дельти Берроуза використано спеціалізований інструментарій faststylometry, який спрощує побудову Delta-корпусу та обчислення відстаней між профілями авторів і цільовими текстами [9], [10], [15]. Це дозволяє зосередитися на інтеграції Delta в ансамбль, а не на низькорівневій реалізації статистичного алгоритму. Збереження навчальних моделей, векторизаторів, калібраторів і агрегованих векторів профілів здійснюється за допомогою бібліотеки joblib, яка оптимізована для серіалізації великих числових об’єктів і забезпечує прийнятний компроміс між швидкістю завантаження та розміром файлів.

Обробка табличних даних (корпусів у форматах CSV, TSV та подібних) здійснювалася за допомогою бібліотеки pandas [20]. Вона використовується як для початкового завантаження текстів і метаданих, так і для формування навчальних і тестових вибірок, агрегації статистичних показників, підготовки проміжних таблиць результатів експериментів. Це дозволяє зберігати єдину структуру даних на всіх етапах – від читання сирого корпусу до побудови підсумкових таблиць з метриками якості.

Генерація пояснюваних звітів реалізована на основі шаблонізатора jinja2 [21]. HTML-структура звіту описується у вигляді шаблону, в який підставляються числові результати верифікації, таблиці з найбільш характерними n-грамами, розподіли функційних слів, текстові пояснення та попередження. Для побудови простих діаграм і графіків у межах звітів використовуються бібліотеки matplotlib або plotly; у типових сценаріях достатньо статичних зображень, які вбудовуються без залежності від

зовнішніх сервісів [22], [23]. У таблиці 3.1 узагальнено основні інструментальні засоби, використані в системі.

Таблиця 3.1 – Основні інструментальні засоби, використані в системі

Категорія	Засоби	Роль у системі
Середовище й обчислення	Python, NumPy, SciPy	Базова платформа, робота з масивами й розрідженими матрицями
Класичне ML	scikit-learn	TF-IDF-векторизація, лінійні моделі, крос-валідація й калібрування ймовірностей
Нейромережеві ембеддинги	sentence-transformers	Отримання векторних подань текстів за допомогою моделей MiniLM
Стилометрія та Delta	faststylometry	Побудова Delta-корпусу та обчислення дельти Берроуза між автором і текстом
Робота з даними	pandas	Табличні корпуси (CSV/TSV), формування вибірок, агрегація результатів експериментів
Збереження артефактів	joblib	Серіалізація моделей, профілів авторів, векторизаторів і калібраторів
Звіти та графіка	jinja2, matplotlib, plotly	Генерація HTML-звітів та побудова базових діаграм і графіків
Інтерфейси	typer / click, streamlit	Командний інтерфейс (train, build-profile, verify, compare-two) та веб-демо застосунок

Командний інтерфейс системи побудовано за допомогою бібліотеки typer або click, які спрощують створення консольних застосунків із підтримкою підкоманд, параметрів і автоматично згенерованою довідкою. Це дозволяє оформити типові операції (навчання моделей, побудова профілів, верифікація, генерація звітів) у вигляді окремих команд, які можуть бути викликані скриптами або інтерактивно. На основі того самого внутрішнього API реалізовано й невеликий демо-застосунок у вигляді веб-інтерфейсу, що працює на базі фреймворку streamlit. Він не є обов'язковою частиною системи, але демонструє можливість інтеграції ядра авторської атрибуції в інтерактивні інструменти. На рисунку 3.1 зображена спрощена схема залежностей між модулями системи та зовнішніми бібліотеками.

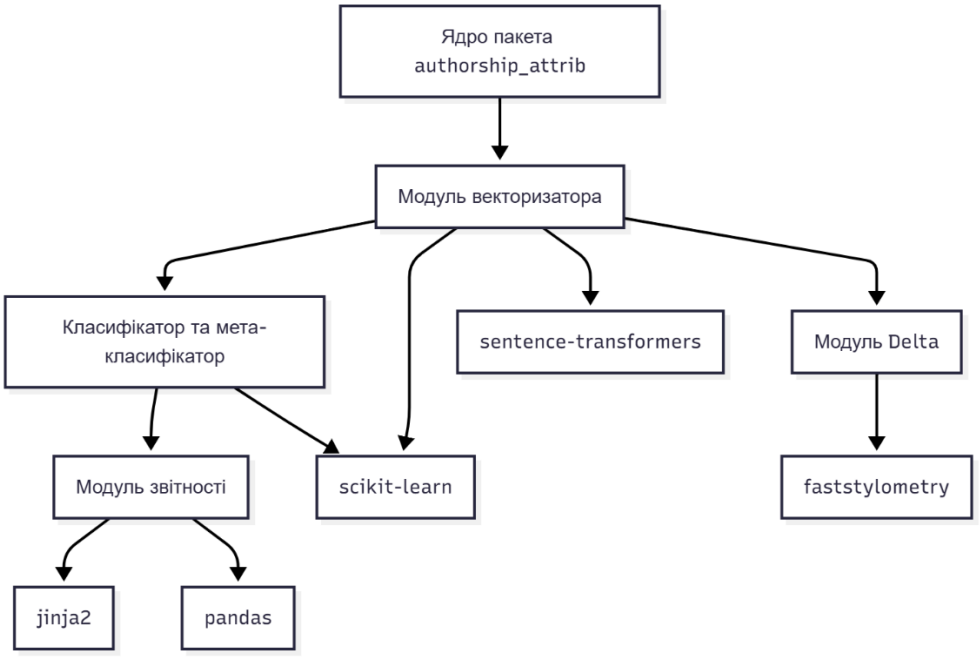

Рисунок 3.1 – Схема залежностей між модулями та бібліотеками

Розробка системи виконувалася у середовищі інтегрованої розробки на зразок PyCharm, що забезпечує підтримку навігації по коду, рефакторингу, налагодження й запуску тестів. Для ізоляції залежностей проєкту використовувалося окреме віртуальне середовище, в якому

встановлювалися всі необхідні бібліотеки. Керування версіями коду здійснювалося за допомогою системи контролю версій Git, що дозволяло фіксувати зміни, повертатися до попередніх варіантів реалізації, а також відокремлювати експериментальні гілки з альтернативними налаштуваннями моделей.

У сукупності вибране середовище розробки та інструментальні засоби забезпечують виконання основних нефункціональних вимог, сформульованих у розділі 2. Система може працювати локально на центральному процесорі без необхідності в графічному прискорювачі, підтримує відтворюваність за рахунок фіксації версій бібліотек і конфігурацій, а також дозволяє поєднувати класичні та нейромережеві підходи в єдиному програмному комплексі. У наступних підрозділах буде розглянуто структуру вихідного коду, організацію корпусу даних і детальні кроки реалізації основних модулів системи.

3.2 Структура вихідного коду та організація програмного проєкту

Програмна реалізація системи авторської атрибуції оформлена як окремий Python-пакет з чітко визначеною внутрішньою структурою. Такий підхід дозволяє використовувати систему як звичайну бібліотеку в інших проєктах, запускати її через командний інтерфейс, а також ізолювати ядро авторської атрибуції від допоміжних скриптів та експериментального коду.

У кореневому каталозі репозиторію розташований власне пакет з кодом системи, файл налаштувань середовища (наприклад, опис залежностей), каталог із прикладами корпусів і каталог експериментів. Пакет містить модулі, які відповідають за попередню обробку текстів, побудову ознак, роботу з методом дельти Берроуза, навчання класичних моделей, побудову та застосування профілів, мета-класифікацію та генерацію звітів. Експерименти та службові скрипти, навпаки, не втручаються у внутрішню структуру пакета, а використовують його як

зовнішню залежність, що сприяє розділенню відповідальностей між «ядром» і прикладним кодом.

Всередині пакета ядро логіки зосереджене в підмодулі, який об’єднує основні компоненти системи: векторизатор, модуль Delta, модуль класичного класифікатора, модуль мета-класифікації, модуль профілів і модуль звітів. Модуль векторизатора містить засоби побудови символьних TF-IDF-представлень, обчислення густих стилометричних ознак і, за наявності відповідної моделі, семантичних ембеддингів. Той самий модуль відповідає за повторне використання навченої конфігурації: він уміє завантажувати збережений стан (параметри TF-IDF, словник, список стилометричних ознак, налаштування попередньої обробки) і застосовувати його до нових текстів. Таким чином, векторизатор є єдиною точкою входу для побудови ознак у всій системі.

Модуль, відповідальний за метод дельти Берроуза, організований так, щоб відокремити «важкий» етап побудови Delta-корпусу від етапу застосування. На етапі підготовки він приймає розмічений корпус, формує необхідні статистики за функційними словами, створює профілі авторів і зберігає всі проміжні об’єкти у вигляді набору файлів. На етапі застосування той самий модуль працює вже як обгортка над збереженими структурами: отримує новий текст, обчислює для нього профіль за тими самими правилами і повертає значення Delta. Це значення далі використовується як одна з ознак у мета-класифікаторі.

Модуль класичного класифікатора містить реалізацію лінійної моделі на основі TF-IDF-ознак, а також засоби її калібрування. Під час навчання він отримує з векторизатора представлення текстів, навчає обрану лінійну модель, застосовує процедуру калібрування ймовірностей і зберігає результат як єдиний об’єкт. Під час застосування модуль приймає вектор ознак для нового тексту чи пари текстів та повертає як класове рішення, так і ймовірнісну оцінку, яка потім може бути використана в складі ансамблю. Важливо, що цей модуль не займається ні побудовою ознак, ні обробкою

профілів; його завданням є виключно навчання та застосування базового класифікатора.

Модуль мета-класифікації працює на рівні вже обчислених показників схожості. Він не взаємодіє безпосередньо з текстами, а очікує на вхід вектор числових ознак, який містить косинусні подібності в різних просторах (TF-IDF, густі стилометричні ознаки, ембеддинги), значення Delta, дивергенції між розподілами функційних слів, різниці в ручних стилометричних характеристиках та інші допоміжні величини. На основі таких векторів модуль навчає лінійну логістичну регресію, а потім, у режимі застосування, перетворює кожен вхідний вектор на підсумкову ймовірність спільного авторства. Завдяки такій ізоляції мета-класифікатор може бути перевчений окремо від базових каналів, наприклад, при зміні складу ознак або при появі нового ембеддингового модуля.

Модуль профілів реалізує логіку роботи з узагальненими описами авторів. Він надає функції для побудови профілю автора за набором текстів, збереження цього профілю у вигляді сукупності бінарних файлів та метаданих, а також для його подальшого завантаження й використання у сценаріях верифікації. При побудові профілю модуль звертається до векторизатора, агрегує TF-IDF-представлення, стилометричні ознаки, ембедингн, формує необхідні статистики для Delta й передає інформацію до модуля мета-класифікації. У режимі верифікації той самий модуль обчислює ознаки для нового тексту, порівнює їх із збереженим профілем, формує повний вектор ознак для мета-класифікатора та повертає підсумкову ймовірність спільного авторства разом із проміжними показниками.

Модуль звітів відповідає за перетворення числових результатів у пояснювану форму. Він приймає профіль, цільовий текст, вектор ознак, підсумкову ймовірність та набір попереджень (наприклад, про надто короткий текст), після чого формує HTML-документ. Внутрішньо цей модуль організовано навколо шаблонів, де кожен блок відповідає за

окремий аспект пояснення: загальний висновок, таблиці схожих і відмінних n-грам, графіки розподілу функційних слів, порівняльні показники стилометричних ознак тощо. Оскільки модуль працює тільки з уже обчисленими даними, зміни у структурі звіту не вимагають втручання в ядро системи.

Зовнішнім шаром над описаними модулями є модуль командного інтерфейсу. Він організований як точка входу, що реєструє окремі команди для навчання моделей, побудови профілів, верифікації авторства, порівняння текстів і генерації звітів. Кожна команда відповідає за розбір параметрів командного рядка, перевірку наявності необхідних файлів (моделей, профілів, конфігурацій), після чого викликає відповідні функції ядра: модуль векторизатора, модуль Delta, модуль класифікатора, модуль профілів і модуль звітів. Завдяки цьому система на рівні сценаріїв використання виглядає як набір самостійних утиліт, що працюють поверх спільного ядра.

Окремо від ядра проєкт містить каталог з експериментами, де зберігаються конфігураційні файли, журнали запусків, проміжні таблиці з результатами та згенеровані звіти. У кожному експерименті фіксуються параметри навчання (діапазон n-грам, вибрані канали ансамблю, параметри розбиття на навчальну і тестову вибірки, випадкові зерна), що забезпечує відтворюваність. Структура вихідного коду та файлової системи відображає цей поділ: ядро з чітко розділеними модулями, тонкий командний інтерфейс та окремий простір для експериментів і звітів.

3.3 Опис корпусу даних

Корпус даних, використаний у цій роботі, складається з коротких україномовних текстів, орієнтованих на реалістичні сценарії інтернет-комунікації. Основна увага приділяється саме тим умовам, у яких задачі авторської атрибуції найбільш складні: невелика довжина повідомлень,

висока варіативність стилю, наявність сленгу, емодзі, хештегів та змішаної мови. Саме такі тексти є типовими для сучасних онлайн-платформ і часто зустрічаються в задачах модерації контенту, цифрової криміналістики та аналізу комунікацій у соціальних мережах. Тому корпус включає тексти, за своєю структурою й лексикою близькі до дописів у соціальних мережах і месенджерах, а не до класичних літературних творів.

Тексти згруповано за авторами, причому в рамках одного автора об’єднано повідомлення, що належать до різних тем і часових відрізків. Така організація дозволяє моделювати реалістичну ситуацію, коли від конкретної особи накопичено набір коротких фрагментів різної тематики, а система має виділити їх спільний стилістичний «слід». Окремі автори можуть мати як відносно вузький тематичний діапазон (наприклад, обговорення одного проєкту чи спільноти), так і дуже розмаїтий набір контекстів, що ускладнює задачу моделювання їхнього стилю. Кожному авторові відповідає анонімізований ідентифікатор; будь-які персональні дані, які дозволяють однозначно ідентифікувати людину поза межами дослідження, вилучені або замінені нейтральними маркерами. Це важливо як з етичних міркувань, так і з погляду стійкості моделі: система має спиратися на стилометричні патерни, а не на окремі унікальні імена, ніки чи посилання.

У межах одного автора окремі повідомлення можуть мати дуже різну довжину: від кількох десятків символів (короткі репліки або реакції) до кількох сотень символів (розгорнуті коментарі або невеликі пости). Для подальшої обробки важливо не лише те, скільки повідомлень є у кожного автора, а й те, як ці повідомлення розподіляються за довжиною. З огляду на вимоги реалізованої системи до мінімальної довжини текстів, при формуванні корпусу враховувалося, що профіль автора має спиратися на сумарний обсяг тексту не менше за кілька тисяч символів, тоді як цільові тексти для верифікації бажано мати довжиною щонайменше кілька сотень символів. Частина надто коротких або майже порожніх повідомлень відфільтровується ще на етапі підготовки даних, аби не вносити зайвий шум

у стилометричні оцінки. На рисунку 3.2 зображено гістограму довжин повідомлень у корпусі Telegram.

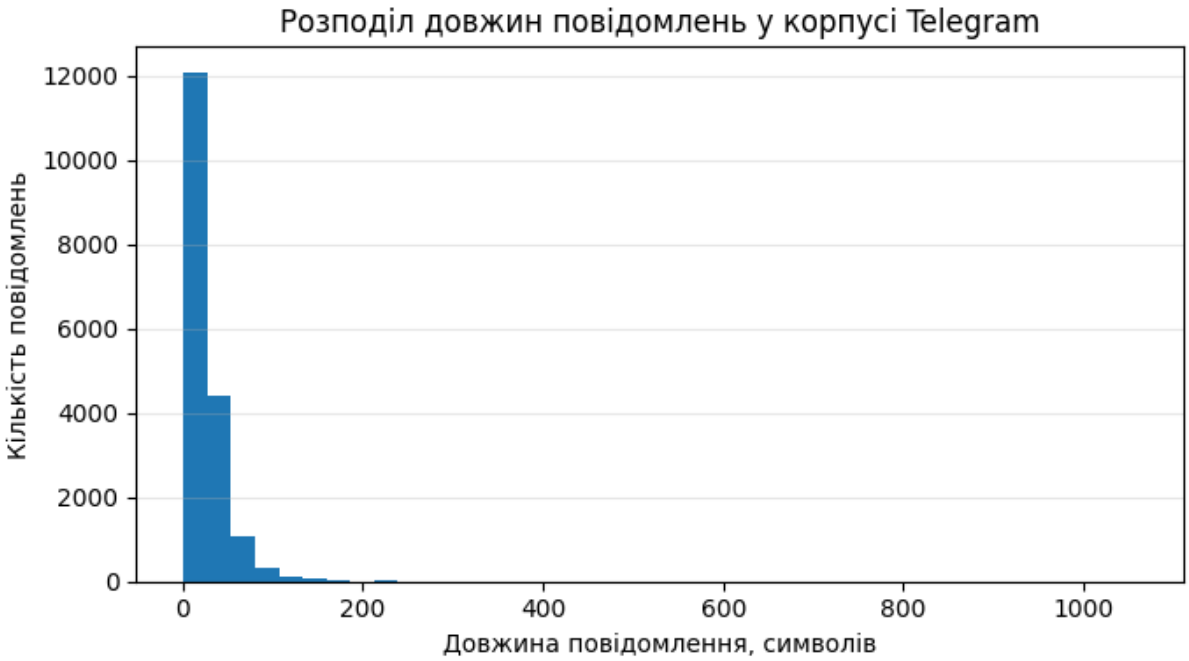

Рисунок 3.2 – Гістограма довжин повідомлень у корпусі Telegram

Логічно корпус організовано у вигляді табличної структури, де для кожного тексту зберігається, як мінімум, ідентифікатор автора та сам текст. Додатково можуть зберігатися службові поля, такі як ідентифікатори джерела або часові мітки, однак вони не використовуються моделлю безпосередньо й можуть бути опущені на етапі експериментів. Така організація спрощує подальший поділ даних на навчальну, валідаційну й тестову частини, а також дає змогу легко будувати різні підвибірки, наприклад для аналізу чутливості якості моделі до довжини текстів.

З точки зору задачі верифікації авторства корпус використовується у двох режимах. По-перше, на його основі формуються профілі авторів: для кожного автора обирається підмножина текстів, які сукупно забезпечують достатню довжину для стабільної оцінки стилю, після чого ці тексти подаються на вхід векторизатору та іншим модулям. По-друге, з того самого

корпусу або з окремо відкладеної частини формуються цільові тексти, які виступають у ролі «кандидатів» для перевірки гіпотези спільного авторства. Частина таких текстів належить тим самим авторам, що й профілі, і використовується як позитивні приклади; інша частина формується із текстів інших авторів і задає негативні приклади. Парами «той самий автор» і «різні автори» згодом користується мета-класифікатор під час навчання.

3.4 Підготовка та попередня обробка даних

Якість роботи системи авторської атрибуції безпосередньо залежить від того, наскільки акуратно підготовлено вхідний корпус. Навіть дуже складний ансамбль моделей деградує, якщо на вхід подати тексти з артефактами, дублікати, фрагменти надто малої довжини або дані, які «підказують» авторство через змістові маркери, а не через стиль. Тому перед навчанням моделей і побудовою профілів виконується кілька послідовних кроків попередньої обробки.

Корпус завантажується у вигляді табличної структури, де для кожного запису зберігаються щонайменше два поля: ідентифікатор автора та текст повідомлення. На цьому етапі здійснюється первинна фільтрація. Видаляються порожні та майже порожні записи, а також тексти, які складаються переважно з технічних маркерів (наприклад, лише посилання чи згадка користувача без жодного «живого» контенту). Окремо перевіряється, чи не дублюються повідомлення: точні дублікати й тривіальні повтори вилучаються, щоб не спотворювати частотні профілі авторів і не завищувати вплив окремих фрагментів.

Далі застосовується серія простих, але важливих нормалізацій. URL-адреси, електронні пошти та згадки користувачів замінюються на службові маркери, які зберігають інформацію про наявність такого елемента, але приховують конкретний вміст. Наприклад, усі посилання незалежно від домену можуть бути зведені до однакового маркера, при цьому навколишні

пробіли й пунктуація зберігаються. Такий підхід мінімізує ризик випадкового «запам’ятовування» конкретних сайтів чи імен, але дозволяє стилометричним ознакам зафіксувати, як часто автор користується гіперпосиланнями і як саме вбудовує їх у текст. Аналогічно нормалізуються й інші технічні об’єкти, які не становлять стилістичної цінності, але можуть нести персональну інформацію.

Особливо обережно система поводиться з емодзі, пунктуацією та регістром. На відміну від традиційних NLP-підходів, які часто «чистять» тексти від смайлів і уніфікують пунктуацію, у задачі авторської атрибуції ці елементи є важливими маркерами стилю. Тому емодзі не видаляються й не замінюються загальними маркерами, а зберігаються як окремі символи, які згодом потрапляють до символьних n-грам і стилометричних ознак. Пунктуація нормалізується лише частково: вирівнюються очевидні технічні артефакти (наприклад, повторювані невидимі символи), але послідовності знаків оклику, питання, комбінації на кшталт «?!» залишаються в тексті в первинному вигляді. Регістр літер також не знижується глобально: для символьних n-грам зберігаються як великі, так і малі літери, оскільки співвідношення між ними входить до стислого стилометричного опису автора.

Після базової нормалізації виконується сегментація тексту. На цьому етапі система розбиває повідомлення на речення за допомогою регулярних виразів, які враховують особливості неформального письма: наявність смайлів наприкінці речення, відсутність пробілу після крапки, використання багатокрапок, поєднання знаків «?!» тощо. Хоча моделі в подальшому працюють переважно на рівні символів, реченнєва структура потрібна для розрахунку агрегованих стилометричних показників: середньої довжини речення, розподілу довжин, частки однослівних або, навпаки, дуже довгих фраз. Ці характеристики входять до writeprints-подібного опису стилю.

Наступний крок стосується обмежень за довжиною текстів та відбору авторів. Занадто короткі тексти – наприклад, односкладові реакції або

відписки довжиною кілька слів – мають дуже слабкий стилометричний сигнал. Якщо включити їх у навчання або верифікацію без фільтрації, вони здатні привнести значний шум, особливо в метриках, що ґрунтуються на частотах символів і функційних слів. Тому запроваджується мінімальний поріг довжини для текстів, які можуть використовуватися як цільові приклади при верифікації. На практиці встановлюється, що тексти коротше за певну кількість символів (наприклад, менше ніж сто символів) не використовуються як окремі приклади, а можуть бути або об’єднані з іншими повідомленнями того самого автора, або повністю відкинуті. Для побудови профілів авторів, навпаки, важливий сумарний обсяг: допускається агрегувати кілька коротких повідомлень, поки їхня сукупна довжина не досягне мінімально необхідного рівня.

Фільтрація застосовується не тільки до текстів, а й до авторів. Автори, для яких доступно занадто мало матеріалу (наприклад, усього кілька дуже коротких повідомлень), не включаються до основного експериментального корпусу, оскільки для них неможливо побудувати стабільний профіль. Таким чином формується підмножина «повноцінних» авторів, для яких є достатня кількість текстів, щоб розділити їх на корпус для профілю та окремі тексти-кандидати для верифікації.

Після відбору авторів і текстів корпус ділиться на навчальну, валідаційну та тестову частини. У задачі верифікації авторства цей поділ має дві складові. З одного боку, необхідно визначити, які тексти використовуються для побудови профілів авторів, а які – як незалежні цільові тексти. З іншого боку, для навчання мета-класифікатора потрібні пари текстів «той самий автор» і «різні автори». Тому для кожного автора випадковим чином відбирається підмножина текстів, які об’єднуються у профіль, а решта текстів розподіляється між навчальною, валідаційною й тестовою частинами таким чином, щоб у кожній з них були як позитивні, так і негативні приклади. На рисунку 3.3 показано, як масив текстів одного

автора ділиться на «профіль» та «кандидатні» тексти, а далі з кількох авторів формується набір позитивних і негативних пар.

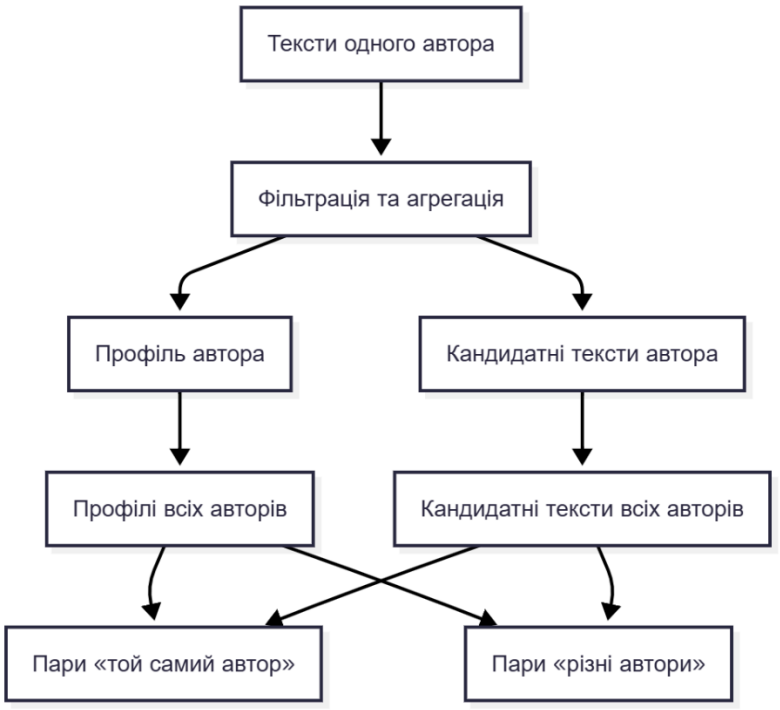

Рисунок 3.3 – Формування профілів авторів та пар текстів для навчання

Формування пар для навчання і тестування мета-класифікатора потребує окремого пояснення. Простий перебір усіх можливих комбінацій текстів одного автора приводить до вибухового збільшення числа пар і, як наслідок, до перекосу датасету на користь авторів з великим обсягом даних. Щоб уникнути цього, використовується контрольоване випадкове вибіркове формування пар. Для кожного автора обмежується максимально допустима кількість позитивних пар, які випадково відбираються з усіх можливих; аналогічно для кожного автора випадковим чином формуються негативні пари, де один текст належить цьому авторові, а другий – будь-якому іншому. Співвідношення позитивних і негативних прикладів обирається таким чином, щоб уникнути сильного дисбалансу й забезпечити стабільне навчання логістичної регресії на рівні мета-класифікатора.

На етапі попередньої обробки здійснюється також анонімізація залишкових персональних ознак, які могли зберегтися в текстах. Імена власні, ніки, специфічні посилання, що однозначно вказують на автора, за можливості або маскуються, або вилучаються з корпусу. Це робиться не тільки з етичних міркувань, а й для того, щоб запобігти «навчанню» моделей на тривіальних маркерах на кшталт власного імені в підписі. Метою є моделювання саме стилю, а не запам’ятовування вмісту.

Результатом описаних процедур є очищений і структурований корпус, у якому для кожного автора визначено придатні для побудови профілю тексти, виділено цільові тексти для верифікації, сформовано навчальні, валідаційні та тестові множини пар «той самий автор» і «різні автори». На цій основі надалі навчаються векторизатор, базові моделі, калібратори та мета-класифікатор. У наступному підрозділі буде розглянуто реалізацію окремих каналів ансамблю та процедури їхнього навчання на підготовленому корпусі [1], [8], [11].

3.5 Реалізація класичних компонентів ансамблю

Класичні компоненти ансамблю реалізовані у вигляді зв’язки векторизатора та окремого класифікатора, які працюють поверх символьних n-грам і густих стилометричних ознак. Їхнім завданням є перетворити сирий текст на числове подання, чутливе до авторського стилю, і далі побудувати на цьому поданні базову модель, здатну відрізняти пари «той самий автор» від пар «різні автори». У реалізації це відповідає класам ClassicVectorizer та ClassicClassifier у модулі aa/model_classic.py, а також набору допоміжних функцій у модулі aa/features.py.

ClassicVectorizer поєднує кілька каналів ознак. Перший і головний канал – символьний TF-IDF для n-грам довжини від трьох до п’яти символів. Він створюється за допомогою фабричної функції char_tfidf_vectorizer з модуля aa/features.py, яка фіксує параметри векторизатора: діапазон n-

грам (3, 5), вимкнене зниження регістру, сублінійну нормалізацію частот термів та мінімальну частоту появи ознаки в корпусі (min_df = 3). Такі налаштування підібрано з урахуванням специфіки коротких текстів: символьні тріграми та п’ятиграми добре вловлюють стилістичні патерни (типові поєднання літер, пунктуацію, емодзі), а збереження регістру дозволяє враховувати індивідуальну манеру використання великих літер. Параметр min_df запобігає надмірному розширенню словника за рахунок випадкових або одиничних помилок, що знижує розмірність простору ознак і робить модель стійкішою.

Другий канал задається набором ручних стилометричних ознак, які обчислюються в тому самому модулі aa/features.py. До них входять функції, що вимірюють частоти різних типів пунктуації, емодзі, співвідношення великих і малих літер, частоту використання функційних слів за спеціальним словником, а також деякі агреговані характеристики на кшталт середньої довжини слова, частки цифр тощо. Ці ознаки формують компактний густий вектор, який описує текст у стилі Writeprints-підходу: не як набір локальних фрагментів, а як сукупність глобальних стилістичних параметрів. Для подальшої інтеграції з TF-IDF ці густі ознаки перетворюються на розріджену матрицю та горизонтально конкатенуються з TF-IDF-представленням за допомогою функції stack_features. У результаті ClassicVectorizer повертає єдину матрицю ознак у форматі розрідженої CSR-матриці, де зібрано як символьні n-грами, так і агреговані стилометричні метрики.

Третій класичний компонент – Delta-пайплайн, також інтегрований у ClassicVectorizer, але працює напівнезалежно. За наявності додаткової залежності faststylometry (extra delta) векторизатор під час fit формує спеціалізований Delta-корпус. Для цього використовується функція default_delta_tokenizer з aa/features.py, яка нормалізує текст до нижнього регістру, розбиває його на токени й виділяє функційні слова, після чого будується частотний профіль кожного автора. Під час transform

векторизатор, окрім TF-IDF та ручних ознак, може повертати матрицю Delta-оцінок, готову для використання в модулі калібрування. Якщо faststylometry недоступна, Delta-канал не активується, а ClassicVectorizer реєструє попередження і працює лише з TF-IDF та ручними ознаками.

Сам ClassicVectorizer реалізує повний цикл роботи з ознаками. Метод fit приймає список текстів (і за потреби – ідентифікатори авторів), навчає TF-IDF-векторизатор, оцінює статистики для ручних стилометричних метрик і, за наявності Delta, формує структури Delta-корпусу. Метод transform застосовує ці перетворення до нового набору текстів і повертає стек ознак у єдиному форматі; fit_transform використовується на етапі навчання. Окремо реалізовано допоміжні методи для роботи з ембеддингами (embedding_vector, has_embedding_channel), а послідовність перетворень і об’єднання каналів ознак у єдину CSR-матрицю показано на рисунку 3.4.

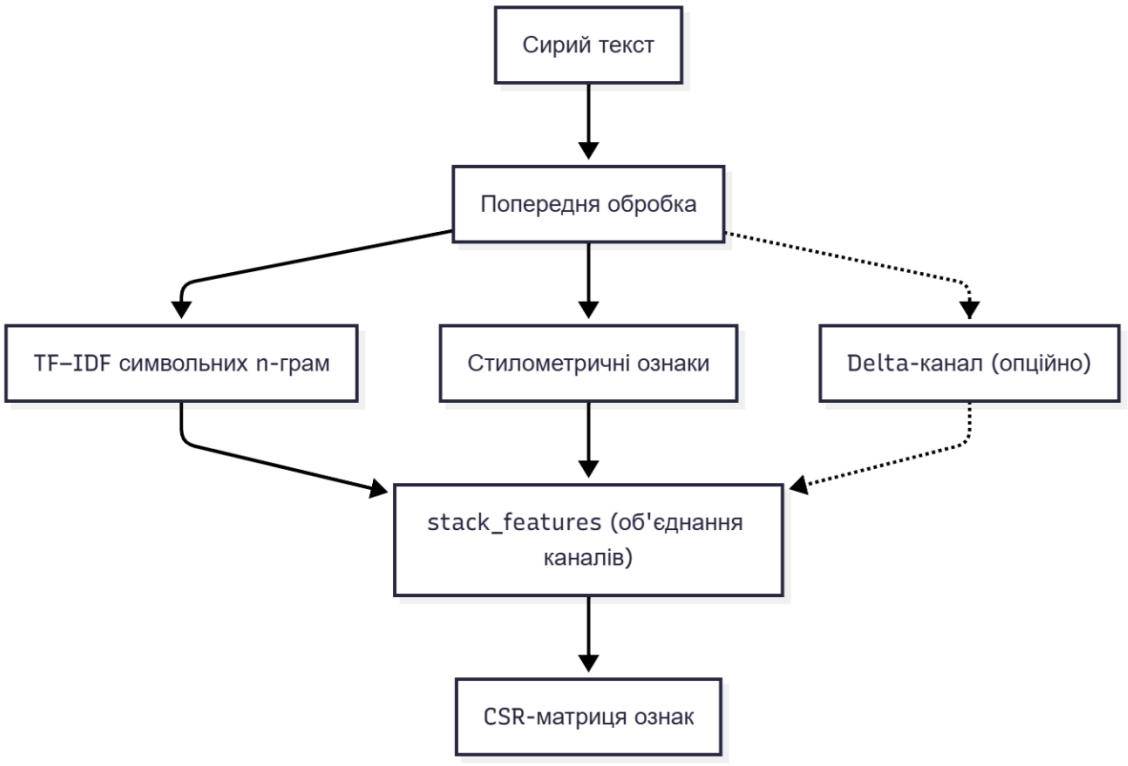

Рисунок 3.4 – Послідовність перетворень у ClassicVectorizer та об’єднання каналів ознак у єдину CSR-матрицю

Другий ключовий елемент класичного блоку – ClassicClassifier, який інкапсулює лінійну модель підтримуючих векторів і процедуру калібрування ймовірностей. Усередині використовується LinearSVC з бібліотеки scikit-learn, що добре працює у високорозмірних просторах, утворених TF-IDF-ознаками та розрідженими матрицями. Оскільки LinearSVC за замовчуванням повертає лише відстань до гіперплощини, ClassicClassifier обгортає його в CalibratedClassifierCV, який на основі додаткової внутрішньої валідації навчає калібрування ймовірностей. У результаті на виході моделі доступні як «жорсткі» рішення, так і оцінки ймовірності для кожного класу. Це критично важливо для ансамблевого підходу: на рівні мета-класифікатора працювати потрібно саме з числовими оцінками, а не лише з бінарними мітками.

Метод fit класу ClassicClassifier приймає матрицю ознак, побудовану ClassicVectorizer, і вектор міток, що відповідають авторам або класам у конкретному завданні. Під час навчання будується лінійний роздільник у просторі ознак, після чого виконується калібрування. Методи predict та predict_proba використовуються відповідно для отримання прогнозованого класу і ймовірнісного розподілу. Внутрішня реалізація враховує можливі помилки та нестачу даних: у випадку, коли калібрування неможливе (наприклад, для надто малого навчального набору), модель може повертати наближені ймовірності, сформовані на основі сигмоїдального перетворення відстані до гіперплощини.

Класичний блок тісно пов’язаний із модулем ознак aa/features.py. Саме тут зосереджена вся логіка побудови TF-IDF-конфігурації, обчислення ручних стилометричних метрик, підготовки даних для Delta-пайплайна та побудови пару ознак для калібрування косинусного методу. Наприклад, для калібрувачного корпусу класичних авторів функція build_calibration_pairs формує пари текстів «той самий автор» і «різні автори», а функція compute_pair_features обчислює для таких пар набір числових характеристик: косинусну подібність у просторі TF-IDF, косинусну

подібність густих стилометричних векторів, значення Delta та його ймовірнісну оцінку, дивергенції між розподілами функційних слів, абсолютні різниці ручних ознак. Усе це надалі використовується в модулі калібрування для побудови окремого калібратора, який працює поверх косинусних мір і Delta.

У сукупності ClassicVectorizer та ClassicClassifier реалізують класичний канал ансамблю: вони перетворюють короткі шумні тексти на числове представлення, що поєднує локальні символьні патерни і глобальні стилометричні параметри, а потім будують на цьому представленні лінійну модель із каліброваними ймовірностями. Delta-пайплайн, інтегрований через модуль features і додатковий канал у векторизаторі, доповнює цю картину, надаючи окремий незалежний сигнал на основі частот функційних слів. У наступному підрозділі буде розглянуто реалізацію нейромережевого компонента на основі семантичних ембеддингів і спосіб інтеграції всіх каналів через калібратор косинусної подібності.

3.6 Нейромережевий компонент та калібрування подібності

Нейромережевий компонент системи реалізовано як додатковий канал ознак, що працює поверх семантичних ембеддингів текстів. Його завданням є надати незалежний сигнал схожості між текстами, який спирається не на символьні патерни або агреговані стилометричні характеристики, а на щільне векторне представлення, сформоване попередньо навченою трансформерною моделлю. Такий канал виконує роль «зовнішнього експерта», що вловлює більш високорівневі закономірності, які класичні моделі можуть не помітити на обмеженому корпусі [3].

У реалізації для обчислення ембеддингів використовується бібліотека sentence-transformers з однією з легких моделей класу MiniLM [13]. Модель завантажується під час ініціалізації відповідного підмодуля й надалі використовується повторно, без перезавантаження на кожен виклик. З

погляду проекту нейромережевий компонент інтегровано як окремий підканал у векторизаторі: клас векторизатора має метод, який приймає список текстів, перетворює їх у послідовності токенів, передає до MiniLM і отримує для кожного тексту вектор фіксованої розмірності, наприклад 384 або 768 компонент. Важливо, трансформер використовується в режимі «застиглих» ваг, що знижує обчислювальні витрати й виключає потребу в GPU. На рисунку 3.5 зображена схема проходження текстів через MiniLM та обчислення косинусної подібності.

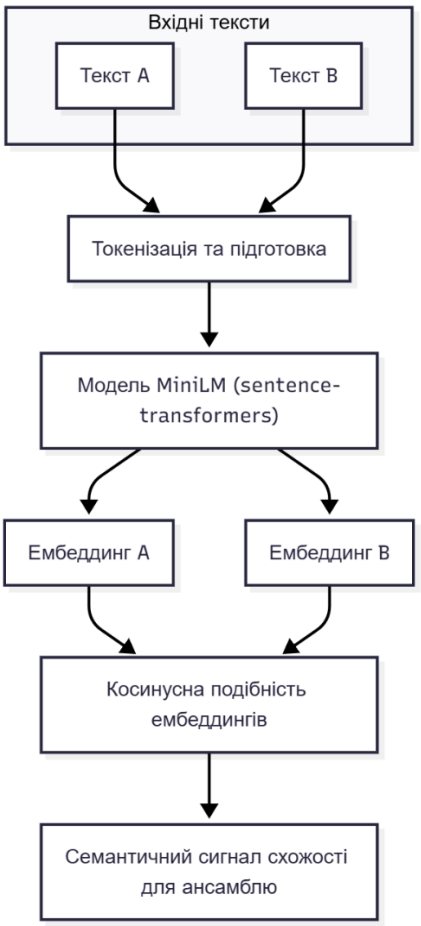

Рисунок 3.5 – Схема проходження текстів через MiniLM та обчислення косинусної подібності

Нейромережевий канал працює не на рівні окремих текстів, а на рівні пар, що відповідає постановці задачі верифікації авторства. Для кожної пари

«профіль автора – текст» або «текст – текст» спочатку обчислюються ембеддинги окремих текстів, після чого розраховується косинусна подібність між цими векторами. Саме це скалярне значення й виступає базовим «сирим» показником від нейромережевого компонента. Додатково для внутрішнього аналізу можуть використовуватися й інші похідні величини, наприклад евклідова відстань між ембеддингами або їх різниця, але в ансамбль передається передусім косинусна подібність як найстабільніша й найінтерпретованіша величина.

Оскільки косинусна подібність не є ймовірністю, а її розподіл суттєво залежить від домену даних, виникає потреба в окремій процедурі калібрування. Для цього використовується підхід, близький до калібрування Delta: на основі набору позначених пар текстів (той самий автор / різні автори) формується вибірка, в якій для кожної пари відомо значення косинусної подібності ембеддингів та цільова бінарна мітка. На цій вибірці навчається проста логістична регресія, яка перетворює значення косинусної подібності на оцінку ймовірності того, що пара належить одному автору. Отримана логістична модель надалі використовується як калібратор ембеддингового каналу: замість «сирої» подібності система працює з узгодженою ймовірнісною оцінкою.

Для навчання калібратора використовуються як пари з основного корпусу коротких текстів, так і пари з калібраційного корпусу класичних авторів. Такий комбінований підхід дозволяє, з одного боку, пристосуватися до реальних умов застосування (короткі, шумні, неформальні тексти), а з іншого – забезпечити більш гладкий і стабільний розподіл прикладів «той самий автор / різні автори» на довших, стилістично чистіших текстах. Логістична регресія при цьому виступає не стільки як «ще одна модель», скільки як функція перенесення: вона відображає проміжні значення косинусної подібності в інтервал [0; 1] у спосіб, узгоджений із реальною частотою позитивних і негативних пар.

Нейромережевий компонент вважається опційним. На рівні реалізації це означає, що підмодуль, відповідальний за MiniLM та ембеддинги, намагається імпортувати необхідні залежності й завантажити модель лише за явного запиту. Якщо залежність sentence-transformers відсутня або середовище виконання не відповідає вимогам (наприклад, надто жорсткі обмеження на пам’ять чи час виконання), система реєструє попередження, відмічає ембеддинговий канал як неактивний і продовжує роботу лише з класичними компонентами. У цьому випадку ансамблевий мета-класифікатор просто не отримує ембеддингові ознаки, а навчається та застосовується на підмножині доступних каналів. Такий дизайн дозволяє не перетворювати нейромережевий блок на «обов’язковий вузол», який блокує всю систему у разі його відсутності.

Окреме місце в реалізації займає модуль, який відповідає за побудову ознак для мета-класифікатора. Саме він об’єднує ембеддинговий канал з іншими каналами схожості в єдиний вектор числових характеристик. Для кожної пари текстів або профіль–текст формується набір ознак: калібрована ймовірність ембеддингового каналу, сире значення косинусної подібності, а за потреби – додаткові статистики на кшталт квадрату відстані або відсікання за порогами. Ці величини додаються до вектора, який уже містить значення Delta, косинусні подібності в просторі TF-IDF, густих стилометричних ознак та інші метрики. Для мета-класифікатора всі ці компоненти виглядають як рівноправні ознаки, і лише в процесі навчання він «вирішує», якою мірою довіряти кожному каналу в різних ситуаціях.

З практичного погляду нейромережевий компонент доповнює класичні методи насамперед у випадках, коли стилістичний сигнал слабкий, а змістові та контекстуальні особливості текстів дають додаткову інформацію. Наприклад, якщо автор послідовно використовує схожі теми, структуру аргументації та типові семантичні шаблони, ембеддинги можуть вловлювати це навіть за умов, коли поверхневі символьні патерни злегка варіюються. Водночас ембеддинги вносять ризик змішування стилю й

тематики: тексти різних авторів, які говорять на одну й ту саму тему, можуть виявлятися занадто близькими в ембеддинговому просторі. Саме тому було принципово важливо не покладатися на нейромережевий канал як на єдине джерело істини, а включити його в ансамбль на рівних з класичними сигналами.

Таким чином, нейромережевий компонент реалізований як окремий канал, що базується на попередньо навченій трансформерній моделі MiniLM та каліброваній косинусній подібності ембеддингів. Він не замінює класичні підходи, а розширює їх можливості, особливо в складних випадках з перефразуваннями й тематичними зсувами. У наступному підрозділі буде розглянуто, як усі канали – класичні, статистичні та нейромережеві – об’єднуються в єдиний ансамблевий мета-класифікатор і як налаштовуються його параметри на підготовленому корпусі пар текстів.

3.7 Ансамблевий мета-класифікатор та об’єднання каналів

У попередніх підрозділах були описані окремі канали системи: класичний TF-IDF з лінійною моделлю, Delta-пайплайн, густі стилометричні ознаки та нейромережевий ембеддинговий компонент. Ансамблевий мета-класифікатор виконує роль «узгоджувача» між цими каналами: він приймає на вхід їхні числові оцінки та на основі спільного аналізу формує підсумкову ймовірність спільного авторства для кожної пари текстів або пари «профіль автора – текст».

З погляду реалізації мета-рівень повністю відокремлений від роботи з сирими текстами. Для нього вихідними є не самі повідомлення, а вектор числових ознак, який будується у спеціалізованому модулі ознак пар. Для кожної пари формується набір величин, що узагальнює інформацію з усіх доступних каналів. До цього вектора входять, зокрема, калібрована ймовірність класичного каналу, косинусна подібність у просторі TF-IDF символьних n-грам, калібрована ймовірність Delta та, за наявності

відповідного калібратора, саме значення Delta, косинусна подібність густих стилометричних векторів, показники схожості розподілів функційних слів, ембеддинговий косинус і його ймовірнісна оцінка. Додатково до цього можуть включатися прості контрольні характеристики: середня довжина текстів у парі, співвідношення їх довжин, індикатор наявності ембеддингового каналу, а також сигнали про гранично короткі тексти. На рисунку 3.6 показано, як виходи окремих каналів (TF-IDF, Delta, стилометричні агрегати, ембеддинги) збираються у вектор ознак і подаються на вхід мета-класифікатору.

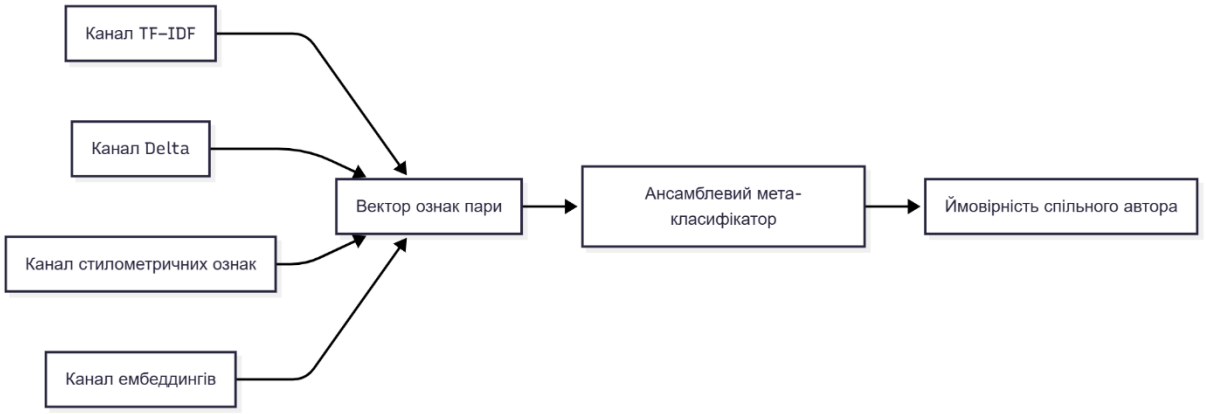

Рисунок 3.6 – Узагальнена схема ансамблевого мета-класифікатора

Мета-класифікатор реалізовано у вигляді лінійної логістичної регресії. Такий вибір продиктовано кількома міркуваннями. По-перше, кількість ознак на цьому рівні порівняно невелика, і немає необхідності вводити складні нелінійні моделі, які вимагали б значно більших обсягів навчальних даних. По-друге, лінійна модель забезпечує прозорість: вагові коефіцієнти при кожній ознаці можна безпосередньо інтерпретувати як міру довіри до відповідного каналу або конкретної метрики. По-третє, логістична регресія природно повертає ймовірність приналежності пари до класу «той самий автор», що узгоджується з постановкою задачі верифікації.

Навчання мета-класифікатора відбувається на спеціально сформованому наборі пар текстів. На цьому етапі використовується корпус, описаний у підрозділах 3.3 та 3.4: тексти для кожного автора діляться на профільну частину та частину для побудови пар. Далі для кожного автора генеруються позитивні пари, де обидва тексти належать цій особі, і негативні пари, де один текст належить авторові, а другий – будь-кому з іншого підмножини. Для кожної пари обчислюється повний набір ознак: викликається векторизатор, класичний класифікатор, Delta-пайплайн, ембеддинговий компонент (якщо він активний), обчислюються косинусні подібності та інші скалярні показники. Результати збираються у матрицю, яка виступає вхідними даними для логістичної регресії, а бінарні мітки «той самий автор / різні автори» задають ціль.

Щоб запобігти перенавчанню й забезпечити узагальнювальну здатність моделі, навчання супроводжується валідацією на відкладеній частині пар. Частина пар, сформованих з окремих текстів, резервується як валідаційна вибірка й не використовується в процесі оптимізації ваг логістичної регресії. Після кожного кроку налаштування параметрів оцінюються базові метрики якості (зокрема, accuracy, F1-міра, площа під ROC-кривою), що дозволяє підібрати регуляризацію моделі та, за потреби, скоригувати склад ознак. Остаточні параметри мета-класифікатора фіксуються після того, як стабілізується якість на валідаційній множині.

Важливою частиною роботи мета-рівня є вибір порогового значення для прийняття рішення. Логістична регресія повертає ймовірність, але для практичного застосування часто потрібен бінарний висновок: «вірогідно той самий автор» або «вірогідно різні автори». Порогове значення обирається не довільно, а на основі аналізу ROC- і PR-кривих на валідаційній вибірці. Розглядаються різні сценарії: максимізація загальної F1-міри, баланс між чутливістю і специфічністю, зсув у бік консервативних рішень (наприклад, уникати хибнопозитивних спрацювань у чутливих застосуваннях). Обране порогове значення фіксується в конфігурації

системи, і надалі всі верифікаційні запити інтерпретуються відносно нього: якщо ймовірність вища або рівна порогу, пара вважається такою, що з великою ймовірністю належить одному автору.

Мета-класифікатор повинен коректно функціонувати й у ситуаціях, коли деякі канали недоступні. Наприклад, якщо в конкретній конфігурації вимкнено ембеддинговий канал або Delta-пайплайн, вектор ознак стає коротшим. Для таких випадків у реалізації передбачено або окремі варіанти навчання мета-моделі для різних комбінацій каналів, або механізм явного заповнення відсутніх ознак значеннями-пропусками й використання відповідної маски. У роботі використовується перший підхід: для основних конфігурацій (повний ансамбль, лише класичні канали) формуються окремі версії калібрувальних моделей. Це дозволяє уникнути неоднозначностей при інтерпретації ваг і забезпечує стабільність поведінки навіть тоді, коли частину каналів вимкнено з міркувань ресурсної економії. На рисунку 3.7 наведено основні групи ознак, що надходять на вхід до двох конфігурацій мета-класифікатора.

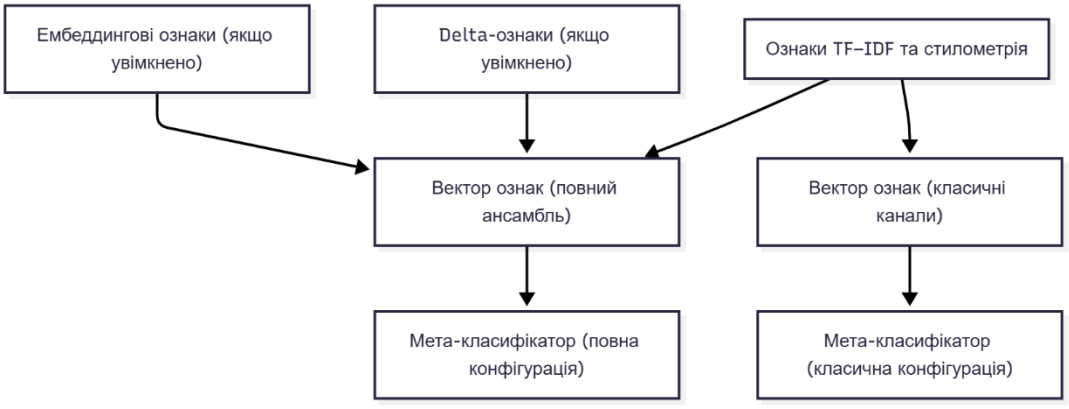

Рисунок 3.7 – Схема ознак, що надходять до мета-класифікатора

З практичного боку мета-класифікатор дозволяє компенсувати слабкі сторони окремих каналів. Наприклад, у випадках, коли Delta дає нестабільні значення через малий обсяг тексту, але TF-IDF і стилометричні агрегати

демонструють високу схожість, логістична регресія може «зменшити вагу» Delta і довіритися іншим ознакам. У протилежній ситуації, коли символьні патерни виявляються схожими, але розподіл функційних слів або ембеддинги явно суперечать гіпотезі спільного авторства, мета-класифікатор, навпаки, зменшить вплив класичного каналу. Таким чином, підсумкове рішення формується не як механічна середня оцінка, а як зважена комбінація сигналів, оптимізована на реальних даних.

У підсумку ансамблевий мета-класифікатор є центральною ланкою, яка перетворює окремі сигнали схожості на єдину ймовірність спільного авторства. Він поєднує виходи класичних, статистичних і нейромережевих компонентів, навчається на реальних парах текстів і дозволяє налаштовувати чутливість системи до різних типів помилок. У наступних підрозділах буде розглянуто реалізацію інтерфейсів системи, процедури побудови профілів та формування пояснюваних звітів, а також наведено результати експериментального оцінювання якості ансамблю.

3.8 Інтерфейси системи та сценарії використання

Інтерфейси системи авторської атрибуції спроєктовано так, щоб вони залишалися мінімалістичними, але достатньо гнучкими для інтеграції в різні робочі процеси. Основним способом взаємодії є командний інтерфейс, орієнтований на запуск типових сценаріїв у терміналі: навчання моделей, побудова профілів, верифікація авторства, порівняння текстів і генерація звітів. Окремі команди можуть поєднуватися у скрипти або використовуватися в середовищах на кшталт Jupyter Notebook, що спрощує відтворюваність експериментів і впровадження системи в існуючі пайплайни аналізу текстів. Інтерактивний веб-демо застосунок слугує додатковим засобом візуалізації можливостей системи та зручного тестування на окремих прикладах без необхідності працювати

безпосередньо з командним рядком. На рисунку 3.8 показано узагальнену схему сценаріїв системи та команд командного інтерфейсу.

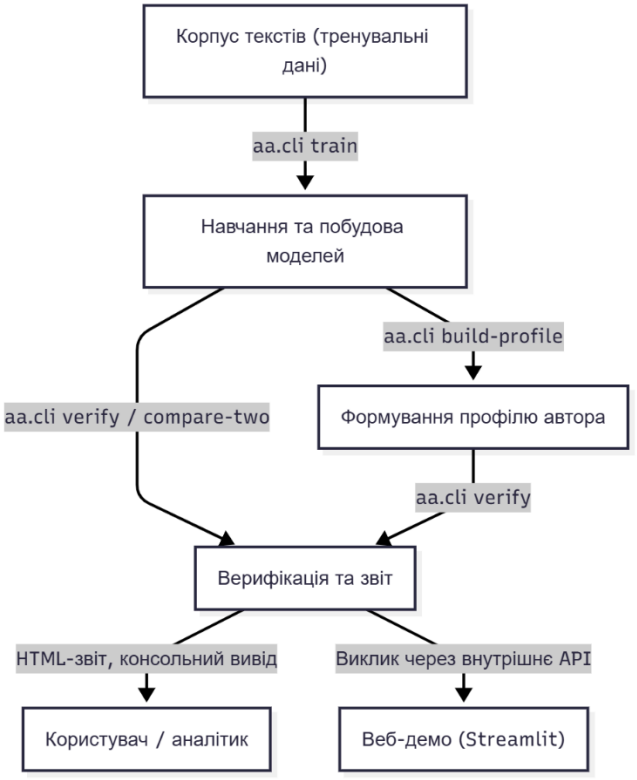

Рисунок 3.8 – Узагальнена схема сценаріїв системи та команд

Командний інтерфейс реалізовано як окремий модуль, який використовує внутрішнє API ядра системи. Кожен сценарій роботи оформлено у вигляді окремої команди з набором параметрів. Користувач може запустити навчання моделей на заданому корпусі, вказавши шлях до файлів з текстами, параметри фільтрації авторів і текстів, а також каталог, у якому будуть збережені результати. У межах одного запуску виконується повний цикл: підготовка корпусу, побудова та навчання векторизатора, класичного класифікатора, Delta-пайплайна, ембеддингового каналу (за наявності), формування калібратора косинусної подібності та навчання мета-класифікатора. Після завершення всі моделі, калібратори й

конфігураційні файли зберігаються в окремому каталозі експерименту, який надалі використовується іншими командами.

Другий тип сценаріїв командного інтерфейсу пов’язаний із формуванням профілів авторів. Користувач задає або каталог з текстами конкретного автора, або табличний файл, де в окремому стовпці зберігаються його повідомлення. Команда завантажує тексти, застосовує процедури попередньої обробки, фільтрує надто короткі фрагменти, пропускає їх через навчений векторизатор і Delta-пайплайн, обчислює всі необхідні стилометричні ознаки та агрегує їх у профіль. Разом із числовими структурами профілю зберігається файл метаданих з описом того, якими моделями, у якій конфігурації та на якому корпусі цей профіль було створено. Надалі профіль можна завантажувати будь-якою іншою командою, за умови сумісності версій.

Третя група сценаріїв пов’язана безпосередньо з верифікацією авторства. Найбільш типова операція – перевірка, чи відповідає новий текст уже наявному профілю автора. Відповідна команда приймає шлях до профілю, текстового файлу або стовпця в табличному файлі, завантажує профіль і моделі, виконує попередню обробку тексту, обчислює ознаки через векторизатор, Delta, стилометричні агрегати й ембеддинговий канал (якщо він активний), формує вектор ознак пари «профіль – текст» та передає його мета-класифікатору. На виході користувач отримує числову оцінку ймовірності спільного авторства, бінарне рішення відносно заданого порогу, а також, за потреби, посилання на згенерований пояснюваний звіт.

Окремим сценарієм є пряме порівняння двох текстів без попередньо збудованого профілю. В цьому випадку команда приймає два текстових файли, застосовує до кожного однаковий ланцюжок попередньої обробки, обчислює ознаки для пари «текст – текст» і далі працює з ними так само, як і зі звичайною парою «профіль – текст». Такий режим зручний для швидких перевірок, коли потрібно попередньо оцінити подібність стилю між двома

фрагментами або протестувати поведінку системи на штучно згенерованих змінених текстах.

Особливістю командного інтерфейсу є його орієнтація на пакетну обробку. Команди дозволяють вказувати не один, а відразу кілька текстів або цілий стовпець у таблиці, а результати верифікацій записуються у вигляді журналу. У журналі для кожного запиту фіксуються ідентифікатор профілю, ідентифікатор тексту, підсумкова ймовірність, бінарне рішення й ключові проміжні показники (наприклад, косинус у просторі TF-IDF, калібрована Delta, ембеддинговий косинус). Такі журнали зручно використовувати для подальшого аналізу, побудови графіків залежності якості від довжини текстів або порівняння різних конфігурацій ансамблю.

Інтерактивний веб-демо застосунок реалізований як тонка оболонка над тим самим внутрішнім API, що й командний інтерфейс. Його основне призначення – показати типовий робочий процес користувачеві, який не працює безпосередньо з терміналом. У рамках застосунку передбачено простий сценарій: користувач обирає із випадаючого списку один із наявних профілів, вставляє текст для перевірки або завантажує файл, натискає кнопку запуску, після чого на екрані відображається оцінка ймовірності, коротке текстове пояснення й вбудована версія HTML-звіту. Усі обчислення при цьому виконуються локально, ті самі моделі й профілі завантажуються з файлової системи, що гарантує однакову поведінку незалежно від того, використовується консоль чи веб-інтерфейс.

Для обох типів інтерфейсів важливо, що вони не вносять додаткової логіки в процес авторської атрибуції. Усі рішення щодо попередньої обробки текстів, побудови ознак, роботи окремих каналів та мета-класифікатора приймаються на рівні ядра системи. Командний рядок і веб-демо лише організовують зручний спосіб передати параметри користувача, запустити послідовність кроків і повернути результат у зручній формі. Такий поділ дозволяє у майбутньому підключати альтернативні

інтерфейси – наприклад, інтегрувати систему в інші програмні комплекси або внутрішні аналітичні платформи – без зміни основних алгоритмів.

У сукупності розроблені інтерфейси підтримують як експериментальне використання (масові перевірки, аналіз якісних характеристик ансамблю), так і прикладні сценарії верифікації авторства, де важливо мати одночасно й числову оцінку, і зрозуміле текстове пояснення, і можливість локальної, конфіденційної обробки даних. У наступному підрозділі будуть підбиті підсумки реалізації та наведено загальні висновки до розділу.

3.9 Висновки до розділу 3

У третьому розділі було деталізовано практичну реалізацію системи авторської атрибуції, спроєктованої в розділі 2, починаючи від вибору інструментів і організації вихідного коду та закінчуючи побудовою ансамблевого мета-класифікатора і користувацьких інтерфейсів. Це дозволило перейти від концептуальної архітектури до конкретних програмних рішень, які можна відтворити, запустити та експериментально оцінити.

Спочатку було обґрунтовано вибір середовища розробки та інструментальних засобів. Використання Python 3.13 у поєднанні з екосистемою бібліотек для машинного навчання, обробки текстів і візуалізації забезпечило єдиний технологічний стек для всіх етапів роботи – від попередньої обробки корпусу до генерації HTML-звітів. Було показано, як поєднання scikit-learn, numpy, scipy, pandas, sentence-transformers, faststylometry, joblib, jinja2 та допоміжних бібліотек для командного інтерфейсу і веб-демо дозволяє задовольнити нефункціональні вимоги: локальну роботу на CPU, відтворюваність експериментів і конфіденційність обробки даних.

Далі було описано структуру вихідного коду та організацію програмного проєкту. Ядро системи оформлено як окремий пакет, у межах якого виділено модулі векторизатора, Delta-пайплайна, класичного класифікатора, мета-класифікації, профілів та звітів. Зовнішній контур утворюють модуль командного інтерфейсу, експериментальні скрипти та веб-демо. Такий поділ забезпечує чітке розділення відповідальностей: алгоритмічна логіка зосереджена в ядрі, а інтерфейсні модулі лише організовують виклики й обробку параметрів. Окремий каталог експериментів із конфігураціями, журналами й звітами гарантує відтворюваність і дозволяє проводити серії дослідів без втручання в ядро системи.

Особливу увагу було приділено опису корпусу даних і процедур попередньої обробки. Було показано, що основний корпус складається з коротких україномовних текстів, близьких до реальних повідомлень у соціальних мережах, доповнений калібраційним корпусом фрагментів творів класичних авторів. Окреслено підхід до анонімізації, відбору авторів і текстів, обмежень за довжиною, а також поділу даних на навчальну, валідаційну та тестову частини. Окремо описано послідовність кроків попередньої обробки: нормалізацію технічних елементів, збереження емодзі й пунктуації як важливих стилометричних маркерів, сегментацію на речення, фільтрацію надто коротких текстів та формування пар «той самий автор / різні автори» для навчання мета-класифікатора.

У підрозділах, присвячених реалізації окремих компонентів ансамблю, детально розглянуто, як саме втілені класичні, статистичні та нейромережеві підходи. Було описано роботу ClassicVectorizer, який поєднує символьний TF-IDF і ручні стилометричні ознаки, а також ClassicClassifier, що будує лінійну модель із каліброваними ймовірностями. Окремо показано, як Delta-пайплайн розділяє етапи побудови частотних профілів і швидкого обчислення Delta для нових текстів. Нейромережевий компонент на основі трансформерної моделі MiniLM реалізовано як

опційний канал, який обчислює ембедінги та косинусну подібність між ними, після чого ця міра приводиться до ймовірнісної шкали через логістичне калібрування на позначених парах текстів.

Центральною ланкою реалізації став ансамблевий мета-класифікатор, що працює поверх вектора ознак, сформованого з виходів усіх каналів: класичного TF-IDF, Delta, густих стилометричних характеристик і ембеддингів. Було обґрунтовано вибір лінійної логістичної регресії як мета-моделі, описано процедури формування навчальної вибірки пар, налаштування регуляризації та вибору порогу для бінарного рішення. Наголошено, що мета-класифікатор може працювати з різними комбінаціями каналів, а його ваги відображають відносний внесок кожного сигналу в підсумкову оцінку ймовірності спільного авторства.

Нарешті, у підрозділі, присвяченому інтерфейсам, було показано, як реалізовані командний інтерфейс і веб-демо застосунок, які використовують одне й те саме внутрішнє API. Описано типові сценарії роботи: навчання моделей, побудова профілів, верифікація текстів проти профілю, пряме порівняння двох текстів, ведення журналів перевірок і генерація HTML-звітів. При цьому підкреслено, що всі обчислення виконуються локально, а інтерфейси не додають нових алгоритмічних рішень, а лише роблять уже реалізовану логіку доступною для користувача.

Узагальнюючи, третій розділ демонструє, що спроєктована в попередньому розділі ансамблева система авторської атрибуції не є абстрактною схемою: вона реалізована в вигляді модульного Python-пакета, здатного працювати з реальними короткими україномовними текстами, будувати профілі авторів, перевіряти гіпотези спільного авторства та формувати пояснювані звіти. На цій основі в наступному розділі буде проведено експериментальне дослідження якості системи: описано методику оцінювання, наведено числові результати для різних конфігурацій ансамблю та проаналізовано сильні й слабкі сторони запропонованого підходу.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Методика дослідження та метрики оцінювання

Експериментальне дослідження спрямоване на перевірку працездатності розробленої системи авторської атрибуції в двох доповнювальних сценаріях. По-перше, оцінюється задача багатокласової класифікації автора окремого короткого повідомлення, яка моделює найскладніший випадок роботи з шумними Telegram-текстами. По-друге, розглядається профільна верифікація, де система має визначити, чи сумісний новий текст із раніше побудованим профілем конкретного автора. Обидва сценарії спираються на один і той самий корпус та один і той самий ансамблевий стек моделей, але по-різному організовують навчальні вибірки, постановку задачі й інтерпретацію результатів.

У багатокласовому експерименті використовувався корпус реальних Telegram-повідомлень кількох авторів, організований у вигляді окремих підкаталогів (по одному на автора). З усіх доступних авторів попередньо відфільтровувалися ті, для кого обсяг даних є недостатнім для навчання стабільної моделі; зокрема, автори з менш ніж 200 повідомленнями не включалися до експерименту. Для кожного автора його повідомлення розбивалися на навчальну, валідаційну та тестову підмножини, причому розбиття виконувалося незалежно в межах кожного класу, щоб уникнути перетікання текстів одного автора між сплітами. Спочатку тексти ділилися на навчальну та проміжну частину у пропорції 70% до 30%, після чого проміжна частина рівномірно розподілялася між валідаційною та тестовою підмножинами (15% і 15% відповідно). Отримані спліти з усіх авторів об’єднувалися в єдині множини train, validation і test, де класовими мітками виступали анонімізовані ідентифікатори авторів.

У цьому ж сценарії порівнювалися дві конфігурації ознак. Базова конфігурація використовує лише класичний канал: символьний TF-IDF у

діапазоні n-грам від трьох до п’яти символів та набір ручних стилометричних ознак, які описують текст глобально (довжини слів і речень, частка пунктуації, великих літер, цифр, емодзі тощо). Розширена конфігурація додає до цього вектора густий ембеддинг MiniLM, обчислений трансформерною моделлю з бібліотеки sentence-transformers. В обох випадках поверх отриманого векторного представлення навчався один і той самий класичний класифікатор (лінійна модель підтримуючих векторів з подальшим калібруванням ймовірностей). Параметри моделей задавалися близькими до стандартних, без агресивного перебору гіперпараметрів, а вибір кращої конфігурації здійснювався за результатами на валідаційній множині. Тестова множина використовувалася тільки один раз, для підсумкової оцінки якості вже відібраної моделі.

Паралельно з багатокласовим експериментом досліджувався сценарій профільної верифікації. У цьому випадку система працює не з класами-авторами безпосередньо, а з попередньо побудованим профілем конкретного автора. На основі всіх доступних Telegram-повідомлень автора orest_borukva формувався профіль: тексти проходили повний ланцюжок попередньої обробки та векторизації, після чого їхні ознаки агрегувалися в стислий опис стилю. Далі цей профіль використовувався для перевірки двох окремих тестових текстів: одного, який належав тому самому автору (файл test_orest.txt), і одного, написаного іншою особою (test_koshys.txt). Для кожного тестового тексту система обчислювала вектор ознак пари «профіль – текст», після чого мета-класифікатор повертав оцінку ймовірності того, що текст належить до профілю.

Щоб проаналізувати вплив довжини тексту на впевненість системи, проводився окремий експеримент із префіксами. Кожен із двох тестових текстів (той самий автор і інший автор) послідовно обрізався до довжин 400, 800, 1200, 1600 і 2000 символів. Для кожного такого префікса повторювалася процедура верифікації проти того самого профілю: будувалися ознаки, запускалися базові моделі та мета-класифікатор, а на

виході фіксувалися оцінка ймовірності спільного авторства і «сирий» показник косинусної подібності між профілем і текстом. Ці значення розглядалися як функція від довжини, що дозволило якісно оцінити, з якого обсягу тексту ансамбль починає давати стабільні рішення.

Для кількісного оцінювання якості багатокласової класифікації використовувалися стандартні метрики: точність класифікації, F1-міра для окремих класів та макроусереднена F1-міра (macro-F1). Точність визначається як частка правильно класифікованих прикладів у тестовій вибірці згідно з формулою (4.1).

$$\text{Точність} = \frac{1}{N}\sum_{i=1}^{N} I\,(\hat{y}_i = y_i), \tag{4.1}$$

де N – кількість прикладів у тестовій множині;

y_i – істинна мітка класу для i-го прикладу;

$\hat{y}_i$ – передбачена мітка;

$I(\hat{y}_i = y_i)$ – індикатор, який дорівнює 1, якщо передбачення правильне, і 0 – інакше.

Для кожного класу k (окремого автора) додатково обчислювалися влучність, повнота та F1-міра. Точність та повнота для класу k визначаються згідно з формулами (4.2) та (4.3).

$$\text{Влучність} = \frac{TP_k}{TP_k + FP_k}, \tag{4.2}$$

де TP_k – кількість вірно віднесених до класу k прикладів;

FP_k – кількість прикладів інших класів, помилково віднесених до k.

$$\text{Повнота} = \frac{TP_k}{TP_k + FN_k}, \tag{4.3}$$

де FN_k– кількість прикладів класу k, помилково віднесених до інших класів. F1-міра для класу k обчислюється як гармонійне середнє між влучністю і повнотою згідно з формулою (4.4).

$$\text{Оцінка } F1 = \frac{2 \cdot \text{Влучність}_k \cdot \text{Повнота}_k}{\text{Влучність}_k + \text{Повнота}_k}. \quad (4.4)$$

Макроусереднена F1-міра визначається як середнє арифметичне F1-показників по всіх K класах згідно з формулою (4.5).

$$\text{Макро-F1} = \frac{1}{K}\sum_{k=1}^{K} F1_k\,, \quad (4.5)$$

де K– кількість класів (авторів).

Саме Макро-F1 використовується як основна узагальнювальна метрика, оскільки вона однаково враховує якість класифікації для кожного автора незалежно від кількості його повідомлень у тестовій множині. Для найкращої конфігурації додатково аналізувалися покласові показники влучність, повнота та F1, а також матриця помилок, яка дозволяє виявити пари авторів, що найчастіше плутаються між собою.

У сценарії профільної верифікації основною вихідною величиною є оцінка ймовірності P(same author), яку повертає мета-класифікатор на основі вектора ознак пари «профіль – текст». У строгому сенсі це також бінарна задача класифікації, і для неї можна було б обчислити ті самі метрики, що й для багатокласового випадку, зокрема accuracy та F1-міру на множині пар «той самий автор / інший автор». Однак у межах цієї роботи профільна верифікація розглядається передусім як демонстраційний кейс. Тому акцент робиться на порівнянні ймовірностей для двох фіксованих текстів (свого автора і чужого) та на аналізі того, як змінюється P(same author) зі зростанням довжини тексту. Додатково для аналізу поведінки моделі

розглядалася сирова косинусна подібність у просторі ознак; її використовували не як самостійну метрику якості, а як допоміжний сигнал, що показує, наскільки калібратор та ансамбль у цілому здатні відокремити стилістично схожі, але належні різним авторам тексти.

4.2 Результати експериментів та їх аналіз

Експериментальне оцінювання системи авторської атрибуції проводилося у двох основних режимах. Перший режим стосувався багатокласової класифікації автора окремого короткого повідомлення в Telegram. Другий розглядав профільну верифікацію, коли система повинна визначити, чи є новий текст сумісним із раніше побудованим профілем конкретного автора. Обидва режими використовують один і той самий корпус і той самий ансамблевий стек моделей, але висвітлюють різні аспекти його поведінки.

У задачі багатокласової класифікації на навчальному, валідаційному та тестовому підмножинах були випробувані дві конфігурації ознак. Базова конфігурація tfidf використовує лише класичний канал: символьний TF-IDF для n-грам довжиною від трьох до п’яти символів у поєднанні з набором ручних стилометричних ознак, що описують текст на глобальному рівні. Розширена конфігурація tfidf_embed додає до цього векторного представлення ще й густий ембеддинг, обчислений трансформерною моделлю MiniLM. В обох випадках поверх конкатенованого вектора ознак навчався один і той самий лінійний класифікатор з подальшим калібруванням ймовірностей.

На валідаційній множині конфігурація tfidf досягла точності близько 0,52 та макроусередненої F1-міри близько 0,41. Для конфігурації tfidf_embed ці показники становили відповідно приблизно 0,53 і 0,42. На тестовій множині обидві конфігурації показали однакову точність, близьку до 0,53, тоді як макроусереднена F1-міра дорівнювала приблизно 0,42 для

tfidf_embed і 0,42 для tfidf з різницею лише в сотих частках. Таким чином, додавання нейромережевого каналу на основі MiniLM дало невелике, але послідовне покращення на валідаційних даних, тоді як на тестовій множині обидві конфігурації виявилися практично еквівалентними.

Для інтерпретації цих значень важливо порівняти їх з наївними базовими стратегіями. За кількості авторів порядку від п’яти до семи випадковий вибір автора мав би точність приблизно від 0,14 до 0,20, а його макроусереднена F1-міра була б на тому самому рівні. Отже, досягнута точність близько 0,53 і макро-F1 близько 0,42 означають, що система працює істотно краще, ніж випадковий або тривіальний класифікатор. З іншого боку, ці значення далекі від «ідеальної» класифікації, що цілком очікувано з огляду на складність задачі: авторство визначається за одним коротким повідомленням, яке саме по собі містить дуже обмежений обсяг стилістичної інформації. У таблиці 4.1 наведено значення точності та макро-F1 для обох конфігурацій tfidf і tfidf_embed на валідаційній та тестовій множинах.

Таблиця 4.1 – Значення точності та макро-F1 для обох конфігурацій

Конфігурація	MiniLM	Accuracy (val)	Macro-F1 (val)	Accuracy (test)	Macro-F1 (test)
tfidf	Ні	0,522	0,407	0,532	0,419
tfidf_embed	Так	0,531	0,422	0,532	0,421

Додатковий аналіз проводився на рівні окремих авторів. Для кожного класу обчислювалися влучність, повнота та F1-міра, після чого формувався окремий набір показників, що дозволяє побачити, наскільки рівномірно система поводиться щодо різних стилів письма. Для частини авторів F1-міра була вищою за середню: як правило, це автори з більшою кількістю повідомлень у корпусі та виразнішим стилем. Інші автори демонстрували дещо нижчу якість, що можна пов’язати або з меншим обсягом даних, або з

тим, що їхній стиль ближчий до стилю інших авторів, і тому межі між класами виявляються менш різкими. На рисунку 4.1 відображені результати оцінки F1 по окремих авторах.

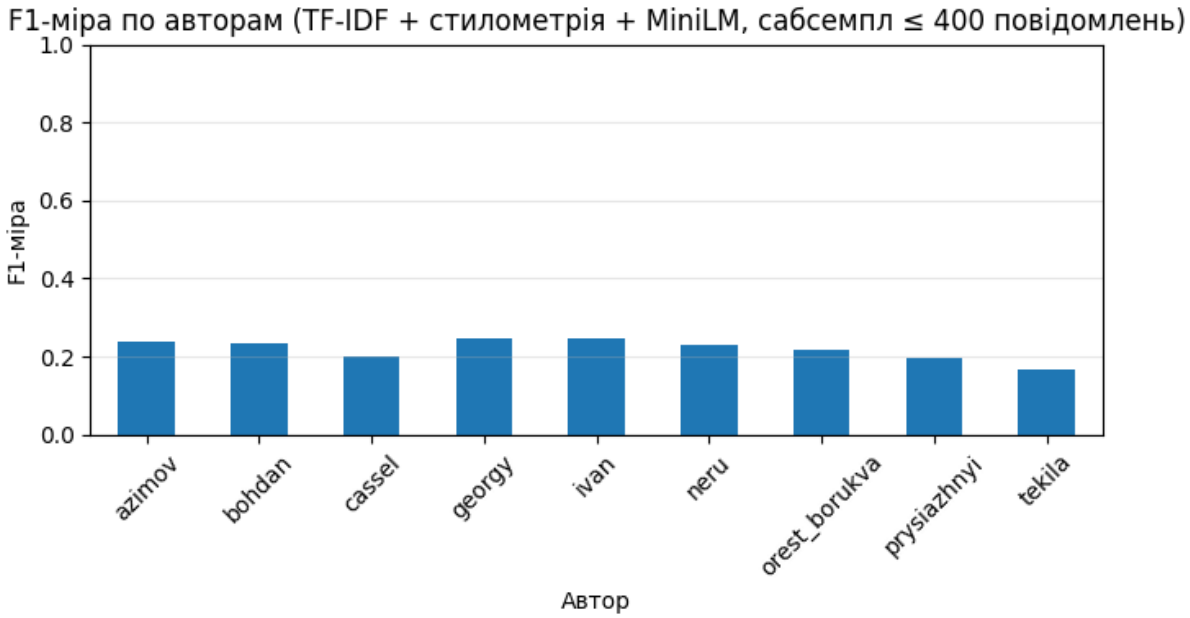

Рисунок 4.1 – Діаграми міри F1 для авторів

Для найкращої конфігурації tfidf_embed була також побудована матриця помилок на тестовій множині. На діагоналі цієї матриці зосереджено правильно класифіковані приклади, поза діагоналлю – помилкові передбачення. Аналіз показав, що найбільше плутанини припадає на пари авторів, які спілкуються в подібних контекстах і використовують схожий сленг та тематичні маркери. При цьому явних «катастрофічних» пар, де система в більшості випадків підміняє одного автора іншим, виявлено не було: навіть для найважчих пар частка правильних класифікацій залишається суттєвою. На рисунку 4.2 зображено теплову карту, де по обох осях відкладено авторів, а інтенсивність кольору в комірці відображає кількість прикладів, віднесених до відповідної комбінації істинного та передбаченого класу.

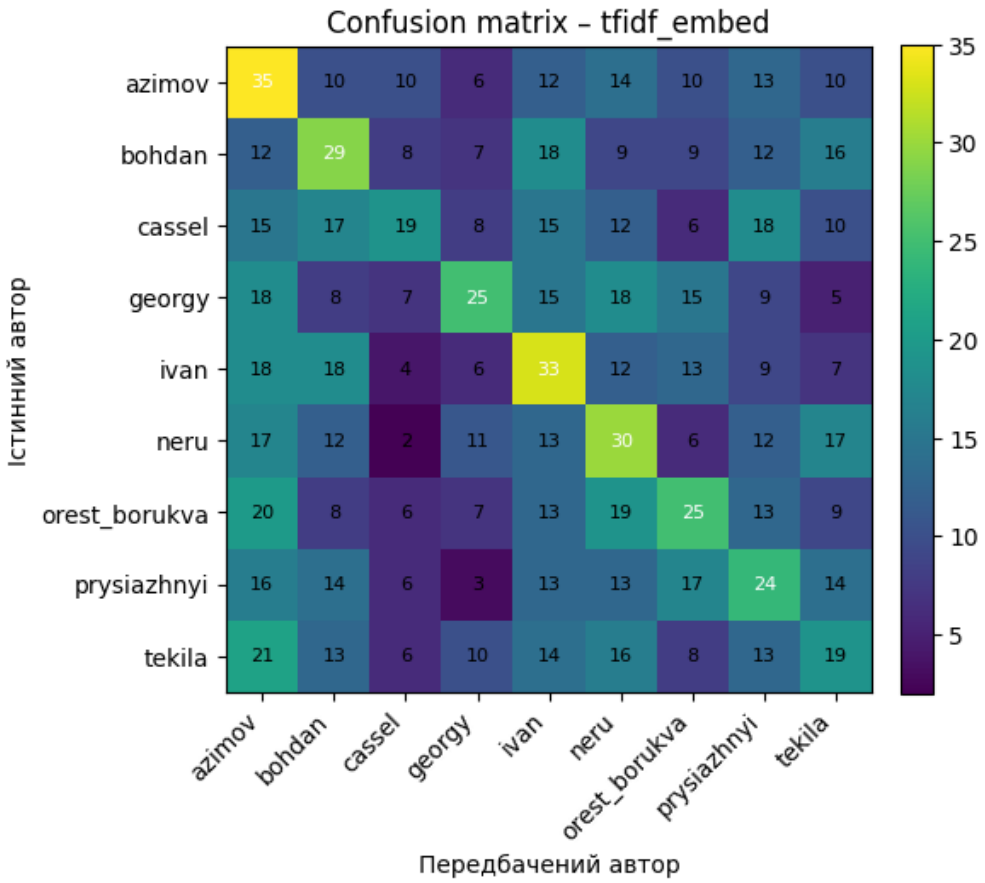

Рисунок 4.2 – Матриця помилок серед авторів

Щоб краще зрозуміти причини обмеженої точності в багатокласовій задачі, окремо аналізувався розподіл довжин повідомлень. Більшість записів у корпусі виявилися дуже короткими: багато повідомлень складаються з кількох десятків символів, тобто однієї–двох фраз або навіть окремих реплік, тоді як значно довші тексти утворюють лише невеликий «хвіст» розподілу. У таких умовах кожне окреме повідомлення несе обмежений стилістичний сигнал, і навіть поєднання символьних n-грам, ручних стилометричних ознак та ембеддингів не може повністю компенсувати брак інформації. Це природно обмежує верхню межу досяжної якості для постановки «одне повідомлення – один прогноз автора». На рисунку 4.3 зображено гістограму довжин повідомлень у корпусі Telegram.

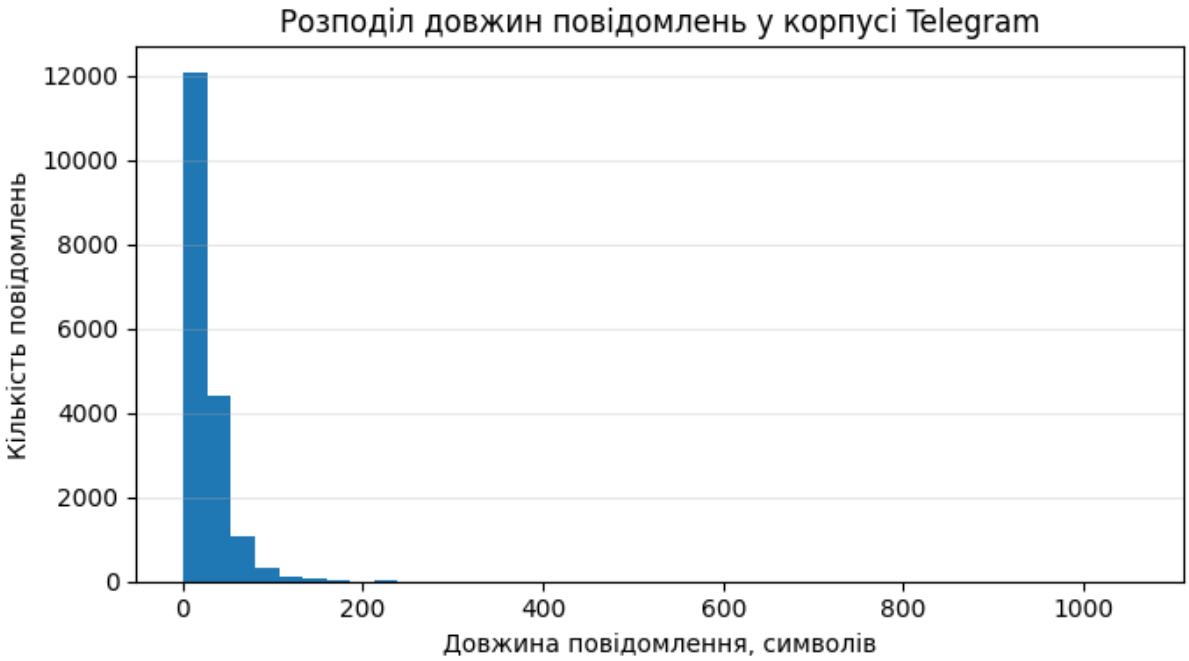

Рисунок 4.3 – Гістограма довжин повідомлень у корпусі Telegram

Паралельно з багатокласовими дослідами аналізувався сценарій профільної верифікації. У цьому режимі для автора orest_borukva був побудований профіль на основі його Telegram-повідомлень, після чого система перевіряла два окремі тексти: один, що належить цьому ж автору (test_orest.txt), та інший, написаний іншим автором (test_koshys.txt). Попри те, що сирий косинус між профілем та обома текстами був дуже високим (для свого автора близько 0,998–0,999, для іншого автора близько 0,94–0,96), мета-класифікатор давав чітко відмінні ймовірнісні оцінки. Для тексту Ореста оцінка P(same author) зазвичай перебувала в діапазоні від 0,67 до 0,90, тоді як для тексту Кошиса – у межах від 0,15 до 0,25. Таким чином, попри високу загальну схожість стилю та тематики в просторі символьних ознак, ансамбль у цілому здатний стабільно відокремлювати власного автора від чужого профілю.

Щоб дослідити залежність верифікації від обсягу тексту, ці самі два тексти додатково розглядалися у вигляді префіксів фіксованої довжини. Для довжин 400, 800, 1200, 1600 і 2000 символів повторювалася процедура

верифікації. У всіх випадках ймовірність P(same author) для тексту Ореста залишалася суттєво вищою за 0,5 (від приблизно 0,64 до 0,91), тоді як для тексту Кошиса – суттєво нижчою за 0,5 (приблизно від 0,15 до 0,25). При цьому значення не зростали монотонно з довжиною: для власного автора ймовірність спершу підвищувалася, а потім дещо знижувалася. Така поведінка є природною для ансамблю, який працює не лише з косинусною подібністю, а й з багатьма додатковими стилометричними ознаками; при додаванні нових фрагментів тексту деякі з цих ознак можуть наближатися або віддалятися від типового профілю автора. Однак головне спостереження полягає в тому, що незалежно від довжини розрив між своїм та чужим автором залишається великим, а рішення системи стабільно відрізняє ці два випадки. На рисунку 4.4 зображено графік зміни ймовірвності авторства залежно від довжини префіксу текста. Криві для свого та чужого автора майже не перетинаються, що добре ілюструє стійке розділення профілю й неавторського тексту.

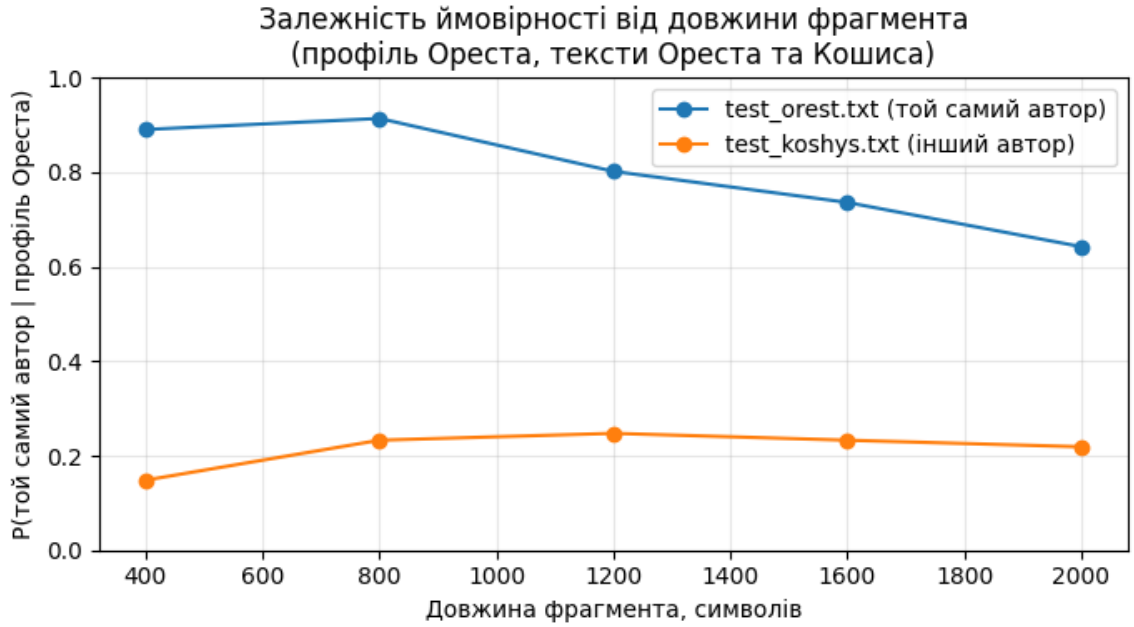

Рисунок 4.4 – Зміна ймовірності залежно від довжини фрагмента

Окремо слід зазначити, що в ході роботи проводилися також попередні експерименти з парною класифікацією «той самий автор / різні автори» на рівні окремих повідомлень без використання профілів. Вони показали, що за великої кількості випадкових пар і високих косинусних значень у просторі символьних ознак класифікатор схильний до виродженої поведінки, фактично віддаючи перевагу гіпотезі «один і той самий автор» для більшості пар. Це проявлялося у високій видимій F1-мірі за рахунок переважання позитивних рішень, але майже випадковій якості за площею під ROC-кривою. З огляду на це, а також на практичну важливість саме профільних сценаріїв, такі постановки не бралися як основний предмет аналізу в цьому розділі й розглядаються радше як попередні спроби, що окреслюють межі можливостей без додаткового профілювання.

У підсумку отримані результати показують, що ансамбль класичних та нейронних ознак забезпечує стійке покращення порівняно з випадковим вибором у складній задачі класифікації автора окремого короткого повідомлення, а в профільних сценаріях демонструє надійне розділення свого автора та чужих текстів уже на фрагментах від кількох сотень символів. У наступних підрозділах буде розглянуто обмеження запропонованого підходу, причини незадовільної поведінки деяких варіантів постановки задачі, а також можливі напрямки подальшого розвитку системи.

4.3 Результати експериментів та їх аналіз

Отримані результати показують, що розроблена система може бути корисною для аналізу коротких україномовних текстів, однак її можливості істотно обмежені як з боку даних, так і з боку обраних алгоритмів та реалізації. У цьому підрозділі узагальнено основні слабкі сторони, які потрібно враховувати при інтерпретації числових результатів і при потенційному практичному використанні системи.

Насамперед слід зазначити, що експерименти виконувалися на доволі вузькому корпусі. Всі тексти походять з одного домену – повідомлень у Telegram – і належать невеликій кількості авторів, які використовують схожий сленг, тематику та спосіб спілкування. Після фільтрації авторів за мінімальною кількістю повідомлень корпус став ще компактнішим, а класи – більш однорідними за жанром і стилістичним контекстом. У таких умовах модель неминуче «вчиться» не лише на власне стилі, а й на доменних особливостях конкретної групи авторів. Це обмежує можливість перенесення отриманих висновків на інші платформи, жанри або навіть на інші групи користувачів у тому самому месенджері.

Другою принциповою обмежувальною обставиною є довжина текстів. Як показав аналіз розподілу довжин повідомлень, переважна більшість прикладів у корпусі є дуже короткими: це одне–два речення або навіть окремі репліки. Кожен такий текст несе лише невелику кількість стилометричної інформації, а випадкові варіації формулювань і теми можуть повністю «перекривати» слабкі індивідуальні патерни автора. У результаті навіть досить складний ансамбль – символьний TF-IDF, ручні ознаки та ембеддинги – стикається з фундаментальною інформаційною межею: за одним коротким повідомленням не можна очікувати високої надійності відновлення авторства. Досягнута точність на рівні приблизно 0,53 і макро-F1 близько 0,42 для багатокласової задачі, з одного боку, свідчить про реальну користь системи порівняно з випадковим вибором, з іншого – демонструє, що для рішень з високою відповідальністю одного повідомлення явно замало.

Третє важливе обмеження стосується калібрування моделей і стабільності поведінки системи в різних постановках задачі. Ранні експерименти з парною класифікацією «той самий автор / різні автори» на рівні окремих повідомлень показали, що при великій кількості пар і високих косинусних значеннях у просторі символьних ознак класифікатор схильний до виродженого режиму, фактично віддаючи перевагу гіпотезі «один автор»

у більшості випадків. Формально це давало прийнятні значення F1-міри, але площа під ROC-кривою та інші калібрувальні індикатори залишалися близькими до випадкового рівня. Цей досвід показує, що на коротких текстах навіть добре налаштовані класичні моделі можуть некоректно поводитися, якщо їх використовувати у невідповідній постановці, а ймовірнісні оцінки потребують окремої, ретельної перевірки.

Профільна верифікація частково пом’якшує цю проблему, оскільки працює з агрегованим профілем автора і дає стабільне розділення між «своїми» та «чужими» текстами, але й тут існують суттєві межі. Профіль потребує достатньо великого й репрезентативного набору текстів, написаних саме цим автором. Якщо профіль побудовано на маленькому наборі або на тематично вузьких фрагментах, рішення системи стають значно менш надійними. Крім того, система не моделює свідоме маскування стилю: якщо автор цілеспрямовано змінює манеру письма, намагаючись сховатися, ансамбль може повернути помилкову низьку оцінку спорідненості навіть для справжнього автора.

Окремо слід виділити обмеження, пов’язані з апаратними ресурсами та реалізацією. У проєкті передбачено повноцінну підтримку методу дельти Берроуза через спеціалізовану бібліотеку, однак у масштабних експериментах з Telegram-корпусом Delta виявилася занадто ресурсоємною. Спроби повної калібровки Delta-корпусу для великої кількості авторів і токенів призводили або до надмірно тривалого часу виконання, або до вичерпання оперативної пам’яті. Через це в усіх підсумкових прогонках Delta була вимкнена, а калібрування цього каналу – відкладене. З формальної точки зору це означає, що одна з ключових концептуальних компонент системи – відстаневий статистичний метод – присутня лише на рівні архітектури, але не пройшла повного експериментального випробування на реальному корпусі коротких текстів. Це зменшує повноту емпіричної оцінки ансамблю.

Подібне, хоча й менш жорстке, зауваження стосується нейромережевого каналу. Використання легких трансформерних моделей класу MiniLM дозволяє обійтися без спеціального обладнання і виконувати обчислення на центральному процесорі, але водночас обмежує спектр доступних архітектур. Більш потужні моделі могли б краще відображати стилістичні нюанси, проте їх використання було би несумісне з вимогами до ресурсної ощадності та автономної роботи системи. Крім того, ембеддинговий канал у роботі використовується без донавчання на специфічному українському корпусі Telegram-повідомлень, тобто модель переносить знання з інших доменів. Це корисно з точки зору узагальнення, але водночас означає, що ембеддинги не оптимізовані саме для задачі авторської атрибуції й частково змішують стилістичні та тематичні особливості текстів.

Ще одне обмеження пов’язане з інтерпретованістю результатів. Попри те що окремі компоненти ансамблю (символьний TF-IDF, агреговані стилометричні ознаки) є досить прозорими й дозволяють виявляти, які n-грамні або стилометричні патерни найбільше впливають на рішення, загальна картина стає менш очевидною, коли їх комбінують із ембеддингами та мета-класифікатором. У профільній верифікації користувачеві надаються HTML-звіти з розподілами ознак і візуалізацією внеску окремих компонент, але повністю «розкласти» підсумкову ймовірність на інтуїтивні складові все одно складно. У практичних сценаріях це може ускладнювати пояснення рішень неспеціалістам або використання системи в контекстах, де потрібна формальна аргументація (наприклад, у правових спорах).

Нарешті, слід згадати про обмеження, пов’язані з залежністю від програмного середовища. Система базується на сучасних версіях бібліотек для машинного навчання й обробки текстів, а також на специфічних версіях Python. Це робить її чутливою до змін у цих бібліотеках: оновлення інтерфейсів або алгоритмів у майбутніх версіях можуть вимагати

додаткових зусиль з підтримки й повторної валідації моделей. Крім того, відсутність окремих необов'язкових залежностей (наприклад, модуля для ембеддингів) змушує систему автоматично вимикати відповідні канали, що знижує якість, якщо не контролювати конфігурацію середовища.

Сукупність перелічених факторів показує, що запропонована система має радше характер дослідницького прототипу, ніж готового до промислового застосування продукту. Вона демонструє принципову можливість поєднання класичних та нейромережевих методів для авторської атрибуції коротких україномовних текстів, але її якість і стабільність істотно залежать від домену, довжини текстів, обсягу профілю, доступних ресурсів і коректності налаштованого середовища. У підрозділі 4.5 ці обмеження будуть використані як відправна точка для формулювання перспектив подальшого розвитку системи, але вже на цьому етапі важливо чітко усвідомлювати межі її застосовності.

4.4 Висновки щодо ефективності розробленої системи

Оцінюючи ефективність розробленої системи авторської атрибуції, доцільно розглядати її поведінку окремо в багатокласовій постановці «одне повідомлення – один автор» та в профільних сценаріях верифікації, а також ураховувати обмеження корпусу й обрані алгоритмічні рішення.

У задачі багатокласової класифікації коротких Telegram-повідомлень система демонструє помірну, але стійку перевагу над випадковими та тривіальними стратегіями. Точність класифікації на тестовій множині становить близько 0,53, а макроусереднена F1-міра – близько 0,42 для обох розглянутих конфігурацій. З огляду на те, що при кількох авторах випадковий вибір давав би точність і макро-F1 на рівні 0,14–0,20, можна стверджувати, що система дійсно вловлює індивідуальні стилістичні відмінності між авторами, навіть коли доступним є лише одне коротке повідомлення. Водночас досягнутий рівень якості явно недостатній для

прийняття рішень у критично важливих контекстах: за одного повідомлення ансамбль скоріше дає попередню гіпотезу щодо автора, ніж безумовний висновок.

Порівняння двох конфігурацій ознак показує, що базовий класичний канал на основі символьного TF-IDF і ручних стилометричних характеристик уже забезпечує значну частину досяжної якості. Додавання нейромережевого каналу на основі ембеддингів MiniLM дає невелике, але послідовне покращення на валідаційній множині, хоча на тестових даних обидві конфігурації виявляються практично еквівалентними. Це свідчить про те, що в умовах обмеженого обсягу даних та дуже коротких текстів класичні стилометричні ознаки залишаються базовим джерелом інформації, тоді як нейромережевий компонент радше виконує роль допоміжного каналу, який може підсилювати модель, але не радикально змінює її поведінку.

У профільній верифікації ефективність системи проявляється значно виразніше. На прикладі профілю автора orest_borukva та двох тестових текстів – свого й чужого – ансамбль стабільно формує високі ймовірності спільного авторства для «свого» тексту (приблизно від 0,64 до 0,90) і низькі для «чужого» (приблизно від 0,15 до 0,25), навіть за умови, що сирі косинусні подібності в просторі ознак є високими для обох випадків. Експеримент із префіксами довжиною від 400 до 2000 символів показує, що розрив між своїм і чужим автором зберігається для всіх довжин, тобто система здатна стабільно розрізняти тексти, якщо має доступ до агрегованого профілю автора й фрагменту тексту середньої довжини. У цьому сценарії ансамбль можна вважати практично корисним інструментом, здатним підтримувати прийняття рішень щодо спорідненості тексту з конкретним автором за умови достатнього обсягу профілю.

Ефективність системи значною мірою визначається самим ансамблевим підходом. Комбінація символьного TF-IDF, ручних стилометричних ознак, ембеддингів та мета-класифікатора дозволяє

компенсувати слабкі сторони окремих каналів і зменшити ризик систематичних помилок, притаманних одній конкретній моделі. Навіть якщо символьні n-грамні представлення виявляються занадто чутливими до тематичних збігів, стилометричні агрегати й ембеддинги додають незалежні сигнали, а логістичний мета-класифікатор вчиться зважувати ці джерела інформації відповідно до їхньої реальної корисності. У результаті система поводиться значно стабільніше, ніж будь-яка її окрема складова.

Водночас потрібно визнати, що потенціал архітектури використано не повністю. Через обмеження ресурсів у підсумкових експериментах не вдалося повноцінно задіяти Delta-пайплайн на Telegram-корпусі, хоча метод дельти Берроуза залишається важливою складовою стилометричного аналізу і природним доповненням до символьних n-грам. Наявність реалізованого, але емпірично не оціненого компонента означає, що отримані числові результати відображають лише частину можливостей закладеного дизайну, а повний потенціал ансамблю ще не розкрито.

З погляду практичних вимог система загалом відповідає сформульованим у розділі 2 нефункціональним критеріям. Вона може працювати локально на звичайному персональному комп’ютері без використання графічних процесорів, підтримує відтворюваність експериментів через збереження конфігурацій і артефактів, а також забезпечує базовий рівень пояснюваності завдяки доступу до окремих каналів та спеціально згенерованим HTML-звітам. Проте її застосування потребує обережної інтерпретації результатів, особливо у випадках, коли аналізується єдине коротке повідомлення, а не сукупність текстів або фрагмент достатньої довжини.

Узагальнюючи, можна зробити такий висновок. У межах поставленої задачі – авторська атрибуція коротких україномовних текстів із використанням ансамблю класичних та нейронних моделей – розроблена система демонструє реальну, але обмежену ефективність. Вона суттєво перевершує випадковий baseline у задачі багатокласової класифікації, проте

залишається далекою від рівня, який дозволив би однозначно встановлювати авторство за окремим повідомленням. Водночас у профільних сценаріях верифікації, де система працює з агрегованим стилем конкретного автора і текстом достатньої довжини, ансамбль виявляється значно надійнішим і придатним для практичного використання як інструмент підтримки прийняття рішень. У подальшому підрозділі ці спостереження будуть використані як відправна точка для формулювання напрямів розвитку системи та можливостей підвищення її точності й стійкості.

4.5 Перспективи подальшого розвитку та вдосконалення системи

Результати експериментів показали, що обрана архітектура є життєздатною, але далеко не вичерпує можливостей як з боку моделей, так і з боку даних. Тому природним наступним кроком є розширення корпусу та поглиблення експериментів у кількох напрямах, які стосуються як якості моделей, так і їх практичної придатності.

Найочевидніший резерв пов’язаний з даними. Поточний корпус охоплює невелику кількість авторів в одному домені, причому більшість текстів є дуже короткими. Розширення корпусу за рахунок нових авторів, а також інших джерел – форумів, довших листувань, блогів – дозволило б краще розділити роль стилю та тематики, оцінити здатність моделі до перенесення між доменами й сформувати більш репрезентативні профілі. Окремим завданням є формування збалансованіших вибірок, де для кожного автора доступні не лише короткі повідомлення, а й фрагменти середньої та великої довжини. Це дало б змогу побудувати залежності якості від обсягу тексту на статистично надійнішій основі, а не лише на окремих кейсах.

Другий важливий напрямок – повноцінне залучення Delta-пайплайна. У поточній роботі метод дельти Берроуза залишився на архітектурному

рівні через обмеження часу виконання та пам’яті під час калібровки на Telegram-корпусі. Подальший розвиток системи міг би включати оптимізацію обчислення Delta, наприклад за рахунок відбору обмеженого набору найбільш інформативних функційних слів, використання розріджених структур зберігання та інкрементальних алгоритмів побудови частотних профілів. Реалістичною стратегією є також поетапна оцінка Delta на менших підкорпусах з подальшою інтеграцією її як додаткового каналу в ансамбль. Окремий інтерес становить вивчення того, чи зможе Delta підсилити систему саме на довших фрагментах тексту, де класичні частотні методи традиційно показують найкращі результати.

Третій кластер перспектив стосується нейромережевих моделей. У роботі використано лише одну, відносно легку трансформерну архітектуру MiniLM без донавчання на специфічному корпусі. Подальші дослідження можуть включати підбір альтернативних моделей для української мови, тонке налаштування ембеддингів на корпусі авторської атрибуції та експерименти з контрастивним навчанням у сіамських або триплетних архітектурах. У такому підході модель навчається безпосередньо зближувати ембеддинги текстів того самого автора та віддаляти ембеддинги текстів різних авторів, що може надати більш виразний сигнал для верифікації. Перспективним виглядає і дослідження символьно-рівневих нейромережевих моделей, здатних працювати з неформальним текстом без попередньої токенізації на слова.

Четвертий напрямок пов’язаний з удосконаленням процедур калібрування та оцінювання. Поточний мета-класифікатор працює з відносно невеликим набором пар і сфокусований переважно на профільних сценаріях. Подальша робота могла б включати побудову більших і структурованіших наборів пар для навчання, балансування співвідношення «той самий автор / різні автори» та систематичне порівняння різних калібрувальних схем. Окрім accuracy і макро-F1 у багатокласовій постановці, доцільно розглядати й інші метрики, більш звичні для

верифікаційних задач: площу під ROC-кривою, точку рівності помилок (Equal Error Rate), детермінантні криві DET. Це дозволить точніше оцінювати якість системи у сценаріях, де важливо контролювати співвідношення хибнопозитивних і хибнонегативних рішень.

П’ятим важливим напрямком є підвищення інтерпретованості. Нині система надає HTML-звіти з розподілами окремих ознак, однак внесок кожного каналу в підсумкову ймовірність залишається непрозорим для кінцевого користувача. Можна розглянути застосування сучасних методів пояснюваного машинного навчання, наприклад локальних апроксимаційних моделей або розкладання внеску ознак у рішенні мета-класифікатора. Це дозволило б будувати більш структуровані пояснення: які саме групи ознак (символьні, стилометричні, ембеддинги, Delta) «голосували» за чи проти гіпотези спільного авторства. Для практичних застосувань, особливо в аналітичних або юридично чутливих контекстах, така прозорість є не менш важливою, ніж сама числова оцінка.

Окремо варто згадати завдання виявлення свідомого маскування стилю та зсувів у часі. Поточна система розглядає стиль автора як відносно статичний, тоді як у реальності він може змінюватися з роками або адаптуватися до різних аудиторій. У майбутніх роботах доцільно розглянути профілі, що залежать від часу, або тематично орієнтовані підпрофілі, які враховують, у яких контекстах автор зазвичай пише. Також перспективним є моделювання «атак», коли автор навмисно змінює манеру письма, і побудова методів, стійких до таких спроб обфускації.

Зрештою, розвиток системи може йти і в бік покращення інтерфейсів та інфраструктури. Це включає автоматизований пошук гіперпараметрів за допомогою пакетів оптимізації, розширення CLI та веб-інтерфейсу для зручнішого запуску експериментів, підтримку журналювання та порівняння кількох конфігурацій ансамблю, а також підготовку стандартизованих форматів експериментальних звітів. Така інфраструктура спростить подальші дослідження й зробить систему більш придатною для

використання в третіх проєктах. На рисунку 4.5 показано основні напрями подальшого розвитку системи.

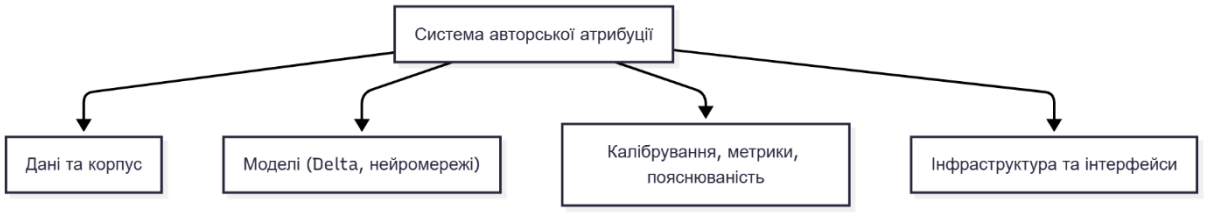

Рисунок 4.5 – Основні напрями подальшого розвитку системи

Таким чином, запропонована система є відправною точкою для подальших досліджень у галузі авторської атрибуції коротких україномовних текстів. Вона демонструє, що поєднання класичних та нейромережевих методів у межах єдиного ансамблю вже зараз дозволяє отримувати практично корисні результати в профільних сценаріях і помітно перевершує випадковий вибір у складній постановці «одне повідомлення – один автор». Подальший прогрес залежатиме від розширення корпусу, більш глибокої інтеграції спеціалізованих методів на кшталт Delta, дослідження сучасних нейромережевих архітектур і вдосконалення процедур калібрування та пояснення результатів.

ВИСНОВКИ

У межах даної кваліфікаційної роботи розроблено та досліджено систему авторської атрибуції коротких україномовних текстів на основі ансамблю класичних стилометричних та нейронних моделей. Спроєктовано та реалізовано Python-пакет authorship_attrib з модулями попередньої обробки тексту, побудови ознак, побудови профілів авторів, верифікації авторства та генерації звітів, а також консольні інструменти для відтворюваних експериментів.

Експерименти на корпусі реальних Telegram-повідомлень показали, що система помітно перевищує наївні базові підходи в задачі багатокласової класифікації, проте окреме коротке повідомлення залишається недостатнім для надійної ідентифікації автора. Найбільш практично придатним виявився сценарій профільної верифікації, де ансамбль моделей забезпечує стійке відокремлення «своїх» і «чужих» текстів за сукупністю повідомлень.

Серед обмежень роботи – порівняно невеликий та однорідний корпус, часткове використання Delta-підходів і функційних слів, а також потреба в кращому калібруванні та пояснюваності результатів. Подальший розвиток системи доцільно пов’язати з розширенням корпусу, повнішою інтеграцією Delta-методів і використанням донавчених трансформерних моделей. Отримані результати мають практичну цінність для досліджень у стилометрії, аналізі соціальних мереж та цифровій криміналістиці й можуть слугувати основою прикладних рішень з верифікації авторства україномовних текстів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Stamatatos E. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*. 2009. DOI: https://doi.org/10.1002/asi.21001 (дата звернення: 20.11.2025).

2. Juola P. Authorship Attribution. *Foundations and Trends in Information Retrieval*. 2008. DOI: https://doi.org/10.1561/1500000005 (дата звернення: 08.12.2025).

3. He X., Lashkari A. H., Vombatkere N., Sharma D. P. Authorship Attribution Methods, Challenges, and Future Research Directions: A Comprehensive Survey. *Information*. 2024. DOI: https://doi.org/10.3390/info15030131 (дата звернення: 20.11.2025).

4. Stamatatos E. Authorship Verification: A Review of Recent Advances. *Research in Computing Science*. 2016. DOI: https://doi.org/10.13053/rcs-123-1-1 (дата звернення: 21.11.2025).

5. Schwartz R., Tsur O., Rappoport A., Koppel M. Authorship Attribution of Micro-Messages. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013). 2013. DOI: https://doi.org/10.18653/v1/D13-1193 (дата звернення: 21.11.2025).

6. Azarbonyad H., Dehghani M., Marx M., Kamps J. Time-Aware Authorship Attribution for Short Text Streams. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2015. DOI: https://doi.org/10.1145/2766462.2767799 (дата звернення: 21.11.2025).

7. Alsanoosy T. Authorship Attribution for English Short Texts. *Engineering, Technology & Applied Science Research*. 2024. DOI: https://doi.org/10.48084/etasr.8302 (дата звернення: 22.11.2025).

8. Eder M., Rybicki J., Kestemont M. Stylometry with R: A Package for Computational Text Analysis. *The R Journal*. 2016. URL: https://journal.r-project.org/articles/RJ-2016-007/ (дата звернення: 22.11.2025).

9. Burrows J. F. “Delta”: A Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*. 2002. DOI: https://doi.org/10.1093/llc/17.3.267 (дата звернення: 22.11.2025).

10. Jannidis F., Pielström S., Schöch C., Evert S. Understanding and Explaining Delta Measures for Authorship Attribution. *Digital Scholarship in the Humanities*. 2017. DOI: https://doi.org/10.1093/llc/fqx023 (дата звернення: 22.11.2025).

11. Abbasi A., Chen H. Writeprints: A Stylometric Approach to Identity-Level Identification and Similarity Detection in Cyberspace. *ACM Transactions on Information and System Security*. 2008. DOI: https://doi.org/10.1145/1330332.1330334 (дата звернення: 23.11.2025).

12. Najafi M., Sadeghian B., Hürriyetoğlu A., Torshizi M. A. K. Text-to-Text Transformer in Authorship Verification. In: CLEF 2022 Working Notes. PAN 2022 – Authorship Verification. CEUR-WS. 2022. URL: http://ceur-ws.org/Vol-3180/ (дата звернення: 24.11.2025).

13. SentenceTransformers. SentenceTransformers Documentation. URL: https://www.sbert.net/ (дата звернення: 24.11.2025).

14. JGAAP. Java Graphical Authorship Attribution Program. Documentation. URL: https://github.com/evllabs/JGAAP (дата звернення: 25.11.2025).

15. Wood T. A. Fast Stylometry [Computer software] (версія 1.0.4). Data Science Ltd., 2024. URL: https://fastdatascience.com/fast-stylometry-python-library (дата звернення: 25.11.2025).

16. Scikit-learn. Scikit-learn 1.5 documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 26.11.2025).

17. Joblib. Joblib: running Python functions as pipeline jobs. Documentation. URL: https://joblib.readthedocs.io/ (дата звернення: 26.11.2025).

18. Harris C. R., Millman K. J., van der Walt S. J. та ін. Array programming with NumPy. *Nature*. 2020. DOI: https://doi.org/10.1038/s41586-020-2649-2 (дата звернення: 26.11.2025).

19. Virtanen P., Gommers R., Oliphant T. E. та ін. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020. DOI: https://doi.org/10.1038/s41592-019-0686-2 (дата звернення: 27.11.2025).

20. McKinney W. Data Structures for Statistical Computing in Python. In: *Proceedings of the 9th Python in Science Conference*. 2010. DOI: https://doi.org/10.25080/Majora-92bf1922-00a (дата звернення: 27.11.2025)

21. Pallets Projects. Jinja Documentation. URL: https://jinja.palletsprojects.com/ (дата звернення: 27.11.2025).

22. Hunter J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*. 2007. DOI: https://doi.org/10.1109/MCSE.2007.55 (дата звернення: 28.11.2025).

23. Plotly Technologies Inc. Plotly Python Graphing Library. Documentation. URL: https://plotly.com/python/ (дата звернення: 28.11.2025).