

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту**

До захисту допущено:
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«__» _____ 20__ р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Моделі класифікації текстів із використанням нейронних
мереж»**

Виконав:
студент IV курсу, групи КІ-22
Мікульський Валерій Валерійович

Керівник:
доцентка кафедри ШІ, д-р філос., доцент
Гуськова Віра Геннадіївна

Консультант з нормоконтролю:
фахівець I категорії кафедри ШІ, к.т.н., доцент,
Комариста Богдана Миколаївна

Рецензент:
професор кафедри ММСА, д.т.н., професор
Петро Іванович Бідюк

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.
Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«15» січня 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Мікульському Валерію Валерійовичу

1. Тема роботи «Розробка моделі класифікації текстів із використанням нейронних мереж», керівник роботи Гуськова Віра Геннадіївна, доктор філософії, доцент кафедри штучного інтелекту, затверджені наказом по НН ПСА від «27» травня 2026 р. № 1944-с.
2. Термін подання студентом роботи «05» червня 2026 року.
3. Вихідні дані до роботи: набори текстових документів українською та англійською мовами з визначеними категоріями, матеріали для навчання та тестування моделі класифікації, результати попередньої обробки текстових даних, дані експериментального оцінювання якості роботи нейромережевої моделі.
4. Зміст роботи: дослідження предметної області автоматичної обробки та класифікації текстової інформації; аналіз сучасних методів машинного навчання, нейронних мереж і засобів обробки природної мови; формалізація задачі віднесення текстів до заданих категорій; вибір інструментів для підготовки даних та побудови моделі; розробка алгоритму попередньої обробки текстів, токенизації та перетворення їх у числове подання; створення нейромережевої моделі класифікації з використанням embedding-шару та

вихідного softmax-рівня; проектування структури програмного застосунку; реалізація вебсистеми для введення, завантаження та класифікації текстових даних; проведення тестування програми та аналіз результатів за метриками accuracy, precision, recall та F1-score.

5. Перелік ілюстративного матеріалу: електронна презентація.

6. Дата видачі завдання «02» лютого 2026 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	21.04.2026	Виконано
2	Підготовка першого розділу	28.04.2026	Виконано
3	Підготовка другого розділу	04.05.2026	Виконано
4	Розробка програмного продукту	12.05.2026	Виконано
5	Підготовка третього розділу	18.05.2026	Виконано
6	Підготовка презентації доповіді	02.06.2026	Виконано
7	Оформлення дипломної роботи	07.06.2026	Виконано

Студент

Валерій МІКУЛЬСЬКИЙ

Керівник

Віра ГУСЬКОВА

РЕФЕРАТ

Дипломна робота: 87 с., 16 рис., 28 табл., 30 джерел.

КЛАСИФІКАЦІЯ ТЕКСТІВ, ОБРОБКА ПРИРОДНОЇ МОВИ, НЕЙРОННІ МЕРЕЖІ, EMBEDDING-ШАР, SOFTMAX, ТОКЕНІЗАЦІЯ, ВЕБЗАСТОСУНОК, TEXTCATEGORY AI

Об'єкт дослідження – процес автоматичної обробки текстової інформації та визначення її тематичної належності.

Предмет дослідження – методи, алгоритми та програмні засоби побудови нейромережевої моделі для класифікації текстових даних.

Метою роботи є розробка моделі класифікації текстів з використанням нейронних мереж та створення програмного продукту для автоматичного розподілу текстових записів за заданими категоріями.

В роботі проаналізовано сучасні підходи до класифікації текстової інформації, методи подання тексту, класичні алгоритми машинного навчання, нейромережеві архітектури та готові бібліотеки для роботи з текстовими даними. Обґрунтовано використання нейромережевої моделі з embedding-шаром, блоком узагальнення ознак, dropout-рівнем і вихідним softmax-шаром для багатокласової класифікації текстів.

Результати роботи – сформовано датасет текстових записів з тематичними категоріями, виконано попередню обробку даних, реалізовано токенізацію, обробку OOV-токенів, padding і truncation, побудовано нейромережеву модель класифікації та створено локальний браузерний вебзастосунок TextCategory AI.

Напрями подальшого розвитку – розширення датасету, підтримка більшої кількості тематичних категорій, використання багатомовних текстових даних, додавання трансформерних моделей, удосконалення механізму аналізу неоднозначних текстів та інтеграція програмного продукту з зовнішніми інформаційними системами.

ABSTRACT

Bachelor's thesis: 87 pages, 16 figures, 28 tables, 30 references.

TEXT CLASSIFICATION, NATURAL LANGUAGE PROCESSING, NEURAL NETWORKS, EMBEDDING LAYER, SOFTMAX, TOKENIZATION, WEB APPLICATION, TEXTCATEGORY AI

The object of the study is the process of automatic processing of textual information and determining its thematic affiliation.

The subject of the study is the methods, algorithms, and software tools for building a neural network model for text data classification.

The purpose of the work is to develop a text classification model using neural networks and create a software product for the automatic distribution of text records into predefined categories.

The work analyzes modern approaches to text information classification, text representation methods, classical machine learning algorithms, neural network architectures, and ready-made libraries for working with text data. The use of a neural network model with an embedding layer, a feature aggregation block, a dropout layer, and an output softmax layer for multiclass text classification is substantiated.

The results of the work include the formation of a dataset of text records with thematic categories, preliminary data processing, implementation of tokenization, OOV token handling, padding and truncation, development of a neural network classification model, and creation of the local browser-based web application TextCategory AI.

Directions for further development include expanding the dataset, supporting a larger number of thematic categories, using multilingual text data, adding transformer models, improving the mechanism for analyzing ambiguous texts, and integrating the software product with external information systems.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ТА МЕТОДІВ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ.....	9
1.1 Актуальність теми та основні сфери практичного використання автоматичної обробки та розподілу текстів за категоріями	9
1.2 Аналіз сучасних підходів та методів розпізнавання тематичної належності текстових даних	12
1.3 Порівняння різних алгоритмів класифікації за основними показниками ефективності	16
1.4 Огляд та порівняльний аналіз готових програмних рішень та бібліотек для роботи з текстами.....	19
1.5 Постановка задачі дослідження з визначенням вхідних даних та критеріїв оцінювання результатів	24
Висновки до розділу 1	27
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТІВ.....	29
2.1 Формалізація задачі автоматичного віднесення текстів до заданих категорій.....	29
2.2 Вибір методів та алгоритмів обробки текстової інформації	34
2.3 Побудова нейромережевої моделі для класифікації текстових даних та опис метрик оцінювання її ефективності	42
2.4 Розробка та проектування архітектури програми та опис основних етапів та логіки роботи.....	49
Висновки до розділу 2	55
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМИ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ	57
3.1 Формування набору текстових даних та виконання попередньої обробки текстової інформації.....	57

3.2 Реалізація моделі для автоматичного розподілу текстів за категоріями	62
3.3 Застосування нейромережі для класифікації на практиці та аналіз отриманих результатів.....	67
3.4 Опис інтерфейсу користувача та детальна інструкція щодо використання програми.....	72
3.5 Тестування системи та оцінка якості її функціонування	77
Висновки до розділу 3	81
ВИСНОВКИ.....	83
ПЕРЕЛІК ПОСИЛАНЬ	85
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....	88

ВСТУП

В сучасних умовах розвитку цифрового середовища значна частина інформації створюється, передається та зберігається у текстовій формі. Новинні повідомлення, звернення користувачів, електронні листи, відгуки, службові документи, навчальні матеріали та повідомлення інформаційних систем формують великі обсяги неструктурованих даних¹. Ручне опрацювання подібних текстів потребує значних часових витрат, залежить від людського фактору та не забезпечує належної швидкості розподілу інформації за тематичними групами.

Актуальність теми кваліфікаційної роботи зумовлена потребою в створенні програмних засобів, здатних автоматично визначати тематичну належність текстових записів. Використання нейронних мереж для класифікації текстів дозволяє перейти від простого пошуку ключових слів до аналізу внутрішніх закономірностей мовного матеріалу. Нейромережеві моделі здатні працювати з числовими представленнями текстів, враховувати повторювані ознаки категорій і формувати прогноз у вигляді ймовірнісного розподілу за класами.

Метою кваліфікаційної роботи є розробка моделі класифікації текстів з використанням нейронних мереж та створення програмного продукту, який забезпечує автоматичний розподіл текстових записів за заданими категоріями.

Для досягнення поставленої мети необхідно виконати певні завдання.

1. Проаналізувати сучасні підходи до автоматичної обробки та класифікації текстової інформації.

¹ Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.).

2. Розглянути класичні алгоритми машинного навчання, нейромережеві архітектури та готові бібліотеки для роботи з текстами.
3. Формалізувати задачу автоматичного віднесення текстових записів до заданих категорій.
4. Обґрунтувати методи підготовки текстових даних, кодування категорій, токенізації та побудови числових послідовностей.
5. Побудувати нейромережеву модель класифікації текстів з використанням embedding-шару та softmax-виходу.
6. Реалізувати браузерний програмний продукт для класифікації текстових записів.
7. Провести тестування програмної системи та оцінити якість її функціонування за метриками accuracy, precision, recall і F1-score.

Об'єктом дослідження є процес автоматичної обробки текстової інформації та визначення її тематичної належності.

Предметом дослідження є методи, алгоритми та програмні засоби побудови нейромережевої моделі для класифікації текстових даних.

Методи дослідження охоплюють аналіз наукових і прикладних підходів до обробки природної мови, методи машинного навчання, нейромережеве моделювання, токенізацію, векторне подання текстів, статистичне оцінювання результатів.

Практичне значення роботи полягає в створенні локального вебзастосунку TextCategory AI, який дозволяє вводити або завантажувати текст, виконувати його попередню обробку, визначати прогнозовану категорію, показувати рівень упевненості моделі, виконувати пакетну обробку та формувати звітні матеріали для аналізу якості класифікації.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ТА МЕТОДІВ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ

1.1 Актуальність теми та основні сфери практичного використання автоматичної обробки та розподілу текстів за категоріями

Інтенсивне зростання обсягів цифрової комунікації перетворило текстову інформацію на один з найважливіших ресурсів сучасного інформаційного суспільства. Повідомлення користувачів, електронні листи, новинні матеріали, відгуки, наукові публікації, службові документи, звернення громадян та записи з інформаційних систем щоденно формують великі масиви неструктурованих даних. Ручне опрацювання значної кількості текстів потребує істотних часових витрат, залежить від суб'єктивного сприйняття виконавця та не забезпечує стабільної швидкості реагування при збільшенні кількості вхідних повідомлень.

Автоматична класифікація текстів належить до ключових задач обробки природної мови, оскільки передбачає присвоєння текстовому об'єкту наперед заданої категорії або мітки. Вказана задача дає змогу перетворити неструктурований мовний матеріал на впорядкований інформаційний ресурс, придатний для пошуку, фільтрації, аналітики, маршрутизації повідомлень та підтримки управлінських рішень [1].

Актуальність теми посилюється через зростання ролі інформаційних сервісів, орієнтованих на швидку взаємодію з користувачем. Для комерційних організацій, класифікація текстів допомагає відокремлювати скарги, запити, позитивні відгуки, питання щодо оплати або доставки. Для державних установ вона корисна під час розподілу звернень за тематикою, пріоритетністю й відповідальними підрозділами. Для освітнього середовища автоматичний аналіз текстів дає змогу групувати навчальні матеріали, перевіряти відкриті відповіді, виявляти тематику звернень студентів. В медіасфері, розподіл новин за рубриками забезпечує зручність навігації та підвищує якість персоналізованої

подачі інформації. Основні сфери практичного використання автоматичного розподілу текстів за категоріями представлено в таблиці 1.1.

Таблиця 1.1 – Основні сфери практичного використання автоматичного розподілу текстів за категоріями

<i>Сфера використання</i>	<i>Характер текстових даних</i>	<i>Практичний результат автоматичної класифікації</i>
Інтернет-медіа та новинні ресурси	Новини, аналітичні матеріали, короткі повідомлення редакцій	Розподіл матеріалів за рубриками, темами й рівнем пріоритетності
Електронна комерція	Відгуки покупців, питання щодо товарів, повідомлення про замовлення	Визначення тематики звернення, виявлення проблемних груп товарів, підтримка персоналізації
Клієнтська підтримка	Заявки, скарги, повідомлення в чатах, електронні листи	Автоматична маршрутизація звернень до відповідного фахівця або підрозділу
Освітні інформаційні системи	Відповіді здобувачів освіти, навчальні матеріали, звернення до адміністрації	Групування матеріалів за темами, первинна перевірка відкритих відповідей, аналіз навчальних запитів
Кібербезпека	Логи, повідомлення про інциденти, описи підозрілих подій	Відокремлення повідомлень за типом загрози, підтримка пріоритезації реагування
Державне управління	Звернення громадян, службові записки, адміністративні документи	Розподіл документів за напрямками діяльності, прискорення документообігу
Медицина та право	Клінічні описи, висновки, договори, правові запити	Тематичне групування документів, пошук релевантних матеріалів, підтримка аналітичної роботи

Автоматична класифікація текстів має міжгалузеве значення та не обмежується вузькою технічною задачею. Її практична цінність полягає в здатності скорочувати тривалість первинного опрацювання інформації, зменшувати навантаження на персонал та забезпечувати однакові правила розподілу повідомлень. За наявності великої кількості текстових записів,

алгоритм може працювати безперервно, а результати класифікації стають підґрунтям для подальшого аналізу тенденцій, ризиків і поведінки користувачів [2].

На відміну від суто словникових або правилкових механізмів, нейромережеві моделі здатні враховувати статистичні закономірності мовних конструкцій, контекст появи слів і приховані зв'язки між ознаками тексту. Сучасна обробка природної мови поєднує лінгвістичні методи, статистичне моделювання, машинне навчання та глибоке навчання, що створює передумови для побудови систем, здатних працювати з великими масивами мовних даних.

В задачах класифікації текст не може безпосередньо подаватися на вхід більшості алгоритмів машинного навчання, оскільки комп'ютерна модель оперує числовими показниками. Перед класифікацією виконується очищення тексту, нормалізація, токенизація, вилучення ознак або побудова векторних показників. Документація `scikit-learn` для роботи з текстовими даними показує типовий приклад перетворення текстових документів на числові ознаки з подальшим навчанням класифікатора, що підтверджує необхідність попередньої підготовки мовного матеріалу перед застосуванням алгоритмів [3].

З точки зору системного аналізу, автоматична класифікація текстів може розглядатися як процес перетворення вхідного потоку мовних даних на структуровані рішення. На вході система отримує текстові повідомлення різної довжини, стилю та тематичного наповнення [4]. Після попередньої обробки, дані переходять в форму числових ознак або векторних показників. На виході формується прогноз належності тексту до певного класу, а якість прогнозу оцінюється за допомогою метрик точності, повноти, F1-міри та матриці помилок.

1.2 Аналіз сучасних підходів та методів розпізнавання тематичної належності текстових даних

Розпізнавання тематичної належності текстових даних належить до центральних завдань обробки природної мови, оскільки від правильного визначення категорії залежить подальше використання повідомлення в інформаційній системі. В загальному вигляді, процес класифікації передбачає перетворення тексту на формалізовану числову інтерпретацію, навчання моделі на заданому датасеті та отримання прогнозу щодо належності нового запису до однієї з наперед визначених груп.

Сучасна практика тематичного розпізнавання охоплює кілька методичних напрямів. Найпростішою основою є механізми, засновані на правилах, де категорія визначається через ключові слова, шаблони або словники предметної галузі. Більш гнучким напрямом вважається статистичне машинне навчання, де текст перетворюється на множину ознак, а класифікатор виявляє залежності між ознаками та мітками класів. Подальший розвиток пов'язаний з нейронними мережами, здатними засвоювати приховані закономірності мовного матеріалу без повного ручного опису ознак [5]. Найвищий рівень контекстного аналізу нині забезпечують трансформерні архітектури, оскільки вони враховують взаємозв'язки між різними частинами тексту за допомогою механізму уваги.

Первинні системи мають важливе історичне та прикладне значення, адже вони забезпечують прозорі правила прийняття рішення. Наприклад, повідомлення про оплату може потрапляти до фінансової категорії за наявності слів «рахунок», «платіж», «транзакція», а звернення про несправність може визначатися через слова «помилка», «збій», «не працює». Перевагою правил є зрозумілість та керованість, однак ефективність швидко знижується за появи синонімів, помилок, скорочень або нових мовних конструкцій.

Статистичне машинне навчання ґрунтується на формуванні ознак, які описують текст в числовій формі. Найпоширенішими способами є Bag of

Words, n-грамні моделі та TF-IDF. Bag of Words фіксує наявність або частоту слів в документі, але не зберігає порядок слів та контекст. N-грамні ознаки частково розширюють модель за рахунок урахування коротких послідовностей слів або символів. TF-IDF оцінює вагу терміну з урахуванням його частоти в документі та поширеності в датасеті, що допомагає виділити тематично значущі слова [6]. Підходи до подання тексту перед класифікацією представлено в таблиці 1.2.

Таблиця 1.2 – Підходи до подання тексту перед класифікацією

<i>Спосіб подання</i>	<i>Принцип формування ознак</i>	<i>Переваги</i>	<i>Обмеження</i>
Правилове подання	Використання ключових слів, шаблонів і словників предметної галузі	Висока пояснюваність рішення, простота контролю	Низька адаптивність до нових формулювань та мовних відхилень
Bag of Words	Фіксація частоти появи слів зі словника корпусу	Простота реалізації, сумісність із багатьма класифікаторами	Втрата порядку слів та змістових зв'язків
N-грамні ознаки	Формування ознак із коротких послідовностей слів або символів	Краще врахування фразових конструкцій	Зростання розмірності простору ознак
TF-IDF	Визначення ваги слова через частоту в документі та рідкісність в датасеті	Виділення тематично важливих термінів	Обмежене відображення контексту
Векторні представлення	Подання слів або документів у вигляді щільних числових векторів	Відображення семантичної близькості	Потреба в якісному корпусі або готовій моделі
Контекстні представлення	Формування вектора з урахуванням оточення слова в конкретному фрагменті	Глибше розрізнення значень і контекстів	Підвищені вимоги до ресурсів

Для класичного машинного навчання найчастіше застосовують наївний баєсівський класифікатор, логістичну регресію, метод опорних векторів, дерева рішень та ансамблеві алгоритми. Наївний баєсівський класифікатор приваблює високою швидкістю навчання та стійкістю до великої кількості ознак. Логістична регресія добре працює з розрідженими TF-IDF-матрицями та дає змогу отримати стабільну базову якість. Метод опорних векторів часто демонструє сильні результати для тематичних корпусів, де класи мають помітні лексичні відмінності [7]. Характеристика алгоритмів тематичної класифікації текстів узагальнено в таблиці 1.3.

Таблиця 1.3 – Характеристика алгоритмів тематичної класифікації текстів

<i>Група алгоритмів</i>	<i>Приклади методів</i>	<i>Сильні сторони</i>	<i>Практичні обмеження</i>
Правилкові алгоритми	Словники, регулярні вирази, шаблони	Прозорість логіки, легке ручне коригування	Слабка робота з неочікуваними формулюваннями
Імовірнісні алгоритми	Naive Bayes	Швидке навчання, добра робота з великим словником	Спрощені припущення щодо незалежності ознак
Лінійні алгоритми	Logistic Regression, Linear SVM	Висока ефективність на TF-IDF-ознаках	Обмежене врахування складних контекстних залежностей
Ансамблеві алгоритми	Random Forest, Gradient Boosting	Здатність моделювати нелінійні залежності	Висока розмірність текстових ознак може ускладнювати навчання
Нейронні мережі	MLP, CNN, LSTM, GRU	Автоматичне засвоєння прихованих закономірностей	Потреба в налаштуванні архітектури та параметрів навчання
Трансформерні моделі	BERT, RoBERTa, DistilBERT	Сильне контекстне подання тексту	Вищі вимоги до апаратних ресурсів

Нейромережеві методи мають особливе значення, оскільки вони дають змогу перейти від ручного конструювання ознак до навчання внутрішніх представлень тексту. Багатошаровий перцептрон може працювати з вже підготовленими TF-IDF-векторами або іншими числовими ознаками. Згортова нейронна мережа придатна для виявлення локальних мовних шаблонів, наприклад характерних поєднань слів в коротких фрагментах. Рекурентні мережі, LSTM та GRU орієнтовані на послідовну природу тексту, оскільки враховують порядок токенів і залежності між фрагментами речення [8].

Окремий напрям формують методи на основі щільних векторних представлень. Моделі Word2Vec, GloVe і fastText дають змогу подати слова в просторі, де змістово близькі одиниці мають близькі координати. fastText додатково враховує підслівні елементи, що корисно для роботи з рідкісними словами, словоформами та помилками введення.

Трансформерні архітектури стали провідним напрямом розвитку класифікації текстів завдяки здатності будувати контекстні представлення. На відміну від традиційних моделей, де вектор слова часто залишається однаковим незалежно від оточення, трансформер формує представлення з урахуванням позиції слова та зв'язків з іншими токенами. BERT було запропоновано як модель двобічного контекстного подання, що навчається на нерозміченому тексті з урахуванням лівого та правого контексту у всіх шарах [9].

Ефективність розпізнавання тематичної належності залежить не лише від алгоритму, але й від попередньої обробки датасету. До важливих етапів належать очищення текстів від службових символів, уніфікація реєстру, токенизація, видалення зайвих елементів, формування словника, обмеження довжини послідовностей та поділ датасету на навчальну, валідаційну та тестову частини. Якість підготовки безпосередньо впливає на здатність моделі відокремлювати категорії, оскільки шум в датасеті, нерівномірний розподіл класів або змішування тематик можуть спотворити результати навчання.

1.3 Порівняння різних алгоритмів класифікації за основними показниками ефективності

Порівняння алгоритмів класифікації текстової інформації має виконуватися не лише за підсумковою часткою правильних відповідей, але й за ширшим набором характеристик, які відображають якість розпізнавання, швидкість навчання, обчислювальну складність та придатність моделі до використання в програмній системі. Для текстових датасетів особливо важливо враховувати нерівномірність класів, оскільки висока загальна точність може приховувати слабе розпізнавання менш представлених категорій. Оцінювання алгоритмів потребує застосування кількох метрик, серед яких найважливішими є accuracy, precision, recall, F1-score, час навчання, швидкість прогнозування та стійкість до шуму в тексті [10].

Accuracy відображає частку правильно класифікованих записів серед всіх прикладів тестової вибірки. Показник є зручним для первинного аналізу, але не завжди достатньо інформативним за нерівномірного розподілу класів. Precision показує, яка частина текстів, віднесених моделлю до певної категорії, справді належить до неї. Recall відображає здатність алгоритму знаходити всі записи відповідного класу. F1-score поєднує precision і recall в одному узагальненому показнику, тому є особливо корисним за потреби збалансованої оцінки. Матриця помилок дає змогу побачити, між якими класами модель найчастіше плутається, що має важливе значення для подальшого вдосконалення архітектури або попередньої обробки датасету [11]. Основні показники ефективності алгоритмів класифікації текстів наведено в таблиці 1.4.

Таблиця 1.4 – Основні показники ефективності алгоритмів класифікації текстів

<i>Показник</i>	<i>Зміст показника</i>	<i>Практичне значення для задачі класифікації текстів</i>
Accuracy	Частка правильних прогнозів серед усіх записів	Дає загальне уявлення про якість моделі, але потребує доповнення іншими метриками
Precision	Частка правильних прогнозів серед текстів, віднесених до певного класу	Важлива за умов, коли помилкове віднесення до категорії має значні наслідки
Recall	Частка знайдених текстів певного класу серед усіх наявних представників класу	Важлива для задач, де небажано пропускати повідомлення певної тематики
F1-score	Гармонійне середнє між precision і recall	Дає збалансовану оцінку якості класифікації
Confusion matrix	Табличне відображення правильних і помилкових прогнозів за класами	Допомагає визначити категорії, між якими виникає найбільше помилок
Час навчання	Тривалість побудови моделі на навчальному корпусі	Визначає зручність експериментів і можливість перенавчання моделі
Час прогнозування	Тривалість визначення класу для нового тексту	Впливає на придатність моделі до інтерактивного використання
Обчислювальні ресурси	Обсяг пам'яті та продуктивність процесора або графічного прискорювача	Визначає можливість запуску програми на звичайному комп'ютері

Модель Naive Bayes часто показує добрі результати на коротких текстах з виразними тематичними словами, оскільки швидко навчається та добре працює з розрідженими ознаками. Logistic Regression є сильним базовим класифікатором для TF-IDF-векторів, оскільки здатна формувати стійку лінійну межу між класами та має помірні вимоги до ресурсів. Linear SVM часто забезпечує вищу якість в просторах великої розмірності, однак потребує ретельного налаштування параметрів регуляризації. Древа рішень та

випадковий ліс менш природно працюють з високорозмірними текстовими ознаками, хоча можуть бути корисними за наявності додаткових структурованих характеристик [12].

Нейронні мережі порівняно з класичними алгоритмами мають іншу логіку навчання. Вони здатні формувати внутрішні представлення тексту, що зменшує залежність від проєктування ознак. Багатошаровий перцептрон може бути застосований до TF-IDF або embedding-векторів, однак він не враховує послідовність слів без додаткового механізму. Згорткові мережі придатні для виявлення локальних мовних шаблонів, а LSTM і GRU краще пристосовані до аналізу порядку токенів. Трансформерні моделі демонструють найвищий потенціал якості завдяки механізму уваги, але вимагають значно більших ресурсів та ретельного контролю навчання.

Вибір моделі залежить від обсягу датасету, довжини текстів, кількості категорій, рівня лексичного перетину між класами та технічних обмежень програмної системи. Для невеликих датасетів, класичні алгоритми часто забезпечують стабільну якість та швидку перевірку гіпотез. Для складніших текстових наборів, де важливу роль відіграють контекст і порядок слів, перевагу отримують нейромережеві архітектури.

Класичні алгоритми мають значну перевагу за швидкістю навчання та простотою відтворення результатів. Naïve Bayes і Logistic Regression можна навчити за короткий час навіть на звичайному комп'ютері, що робить їх зручними для побудови контрольного рівня. Linear SVM часто потребує трохи більше часу, але може дати кращі межі між класами. Вказані моделі добре поєднуються з TF-IDF, що спрощує підготовку експерименту та інтерпретацію впливу окремих термінів.

Нейромережеві моделі є складнішими з погляду налаштування, але їх перевага полягає в здатності навчатися на послідовностях токенів та будувати внутрішні векторні представлення, які враховують закономірності датасету [12].

Порівняння алгоритмів варто організувати за принципом поступового ускладнення. Спочатку може бути побудована базова модель на основі TF-IDF і Logistic Regression або Naive Bayes. Далі доцільно реалізувати нейромережеву модель з embedding-шаром та класифікаційним блоком. Після навчання обидві моделі можуть бути оцінені на однаковій тестовій вибірці за accuracy, precision, recall і F1-score [13].

Порівняння різних алгоритмів класифікації демонструє, що ефективність моделі не можна оцінювати за одним числовим показником. Висока accuracy має значення лише за умови прийнятних precision і recall для всіх класів. Швидкість навчання важлива для експериментальної роботи, а швидкість прогнозування визначає придатність програмного продукту для інтерактивного режиму. Пояснюваність результатів має значення під час аналізу помилок, тоді як здатність працювати з контекстом є ключовою перевагою нейромережових і трансформерних моделей.

1.4 Огляд та порівняльний аналіз готових програмних рішень та бібліотек для роботи з текстами

Готові програмні рішення для роботи з текстовою інформацією включають хмарні сервіси, бібліотеки машинного навчання, фреймворки глибокого навчання та спеціалізовані інструменти для обробки природної мови [14]. Їх використання дає змогу скоротити час розробки, застосувати перевірені механізми векторизації, навчання та оцінювання моделей, а також, порівняти власну реалізацію з відомими технологічними засобами [15].

Готові рішення можна умовно поділити на дві великі групи. До першої належать хмарні сервіси, які надають готові інструменти аналізу тексту через вебінтерфейс або API [16]. До другої групи належать бібліотеки та фреймворки, які встановлюються в середовищі розробки та дають можливість створити

власну модель з повним контролем над архітектурою, параметрами навчання та обробкою датасету [17].

Одним з відомих хмарних рішень є Google Cloud Natural Language [18]. Функція Content Classification аналізує документ та повертає список категорій, які відповідають змісту тексту; для виконання операції використовується метод `classifyText`. Сервіс орієнтований на інтеграцію з іншими інформаційними системами через API, тому його можна розглядати як приклад інструменту для автоматичного тематичного розподілу документів [19]. Сторінка сервісу Google Cloud Natural Language для класифікації текстового контенту наведена на рисунку 1.1.

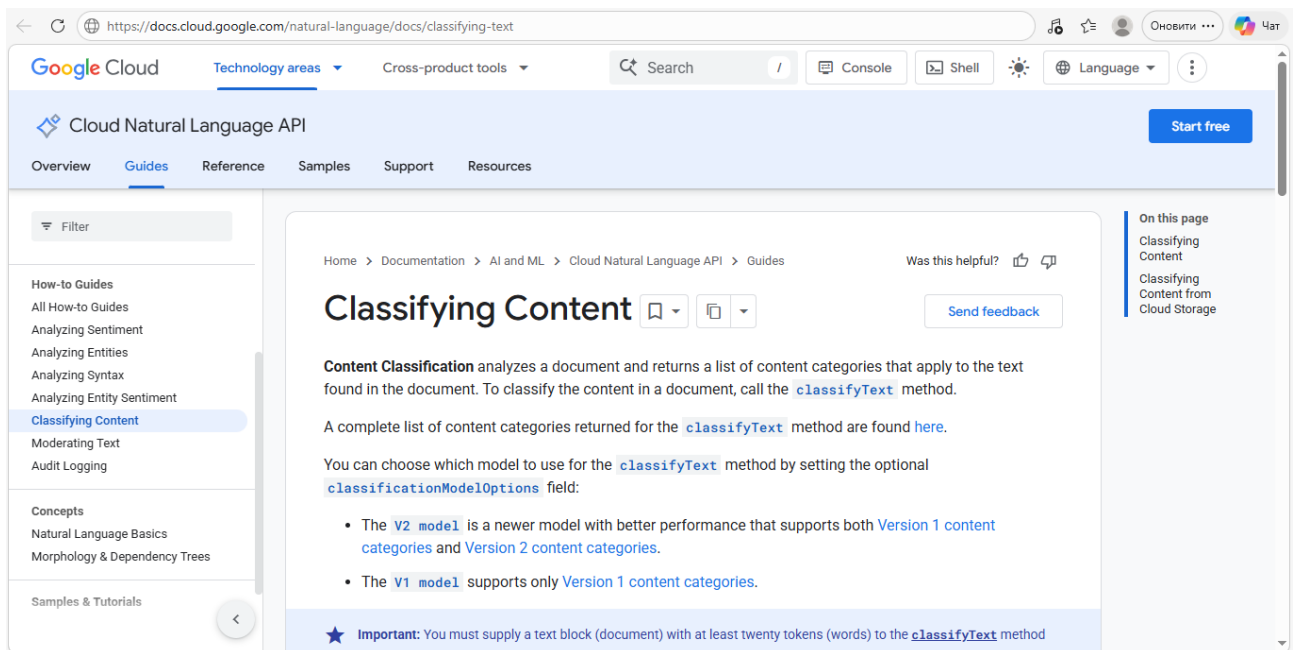


Рисунок 1.1 – Сторінка сервісу Google Cloud Natural Language для класифікації текстового контенту

Amazon Comprehend також належить до готових хмарних NLP-сервісів. Користувач може застосовувати попередньо навчені моделі або навчати власні моделі для класифікації та розпізнавання сутностей, а доступ до можливостей аналізу документів надається через консоль Amazon Comprehend або API. Функція `custom classification` дає змогу організовувати документи за

визначеними користувачем класами після навчання власного класифікатора [20]. Сторінка сервісу Amazon Comprehend з можливостями класифікації документів наведена на рисунку 1.2.

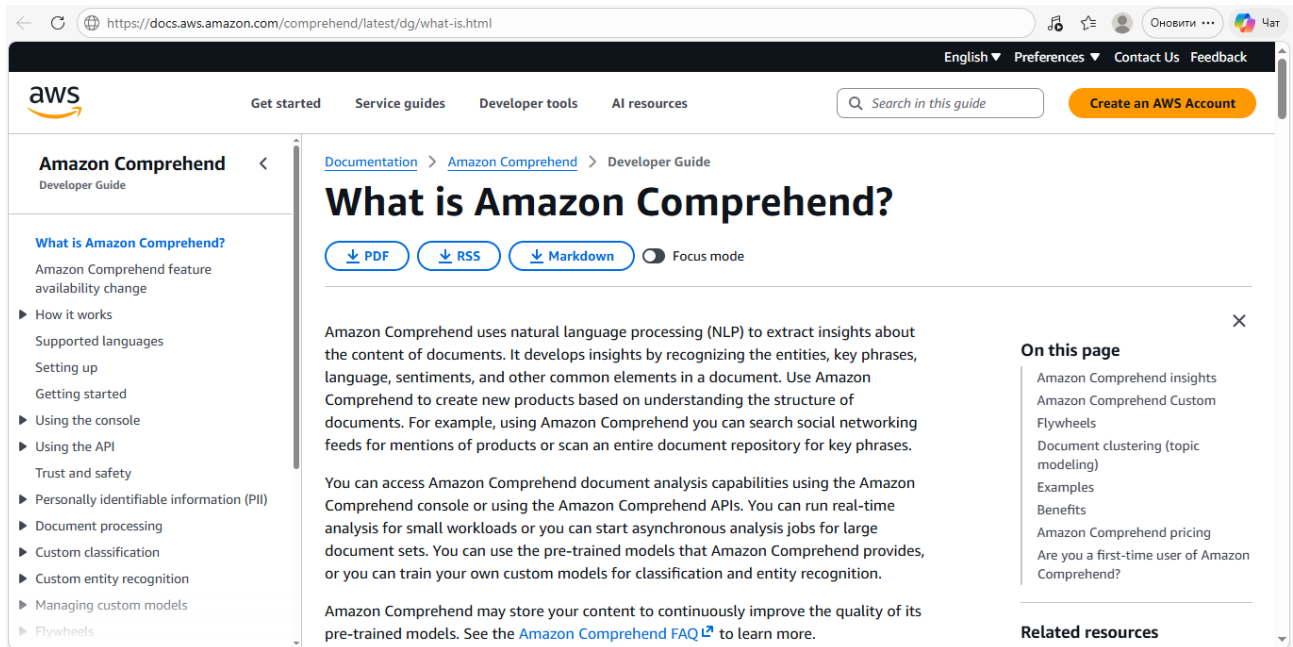


Рисунок 1.2 – Сторінка сервісу Amazon Comprehend з можливостями класифікації документів

Серед бібліотек для локальної реалізації важливе місце посідає `scikit-learn`. Документація `scikit-learn` містить окремий навчальний розділ `Working With Text Data`, де розглянуто завантаження текстових документів, виділення ознак, навчання лінійної моделі, побудову `pipeline`, оцінювання результатів та налаштування параметрів.

Бібліотека `sraSu` орієнтована на обробку природної мови в Python. Дана бібліотека підтримує токенізацію, розпізнавання іменованих сутностей, частиномовне тегування, залежнісний аналіз, векторні представлення та `text classification`. Окремий компонент `TextCategorizer` прогнозує категорії для документа та має два варіанти: `textcat` для взаємовиключних міток і `textcat_multilabel` для багатоміткової класифікації.

TensorFlow і Keras є одними з найпоширеніших засобів для побудови нейронних мереж. TensorFlow будує класифікатори на основі звичайних текстових файлів, навчання моделі для аналізу тональності та можливості адаптації прикладу до багатокласової класифікації.

PyTorch є альтернативним фреймворком глибокого навчання, який часто використовується в дослідницьких проєктах завдяки гнучкому опису моделей та зручному керуванню процесом навчання. PyTorch працює з текстовими задачами, словниками, токенізаторами та наборами даних, а навчальний приклад PyTorch демонструє навчання текстового класифікатора SST-2 з використанням попередньо навченої XLM-RoBERTa-моделі.

Hugging Face Transformers посідає окреме місце серед інструментів для роботи з текстами, оскільки надає доступ до великої кількості попередньо навчених моделей та підтримує донавчання для задач класифікації послідовностей. Бібліотека Transformers подається як фреймворк для сучасних моделей машинного навчання в тексті, комп'ютерному зорі, аудіо, відео та мультимодальних задачах. Порівняння бібліотек і фреймворків для реалізації класифікації текстів наведено в таблиці 1.5.

Порівняння показує, що вибір інструменту має залежати від мети конкретного етапу роботи. Для попереднього аналізу та формування базового рівня якості потрібно застосувати scikit-learn, оскільки бібліотека швидко реалізує схему TF-IDF і класичний класифікатор. Для поглибленої лінгвістичної обробки можна використати spaCy, особливо коли потрібно виконати токенізацію або провести додатковий аналіз структури тексту. Для побудови власної нейромережевої моделі найбільш обґрунтованим варіантом є TensorFlow/Keras, оскільки він поєднує доступність коду, підтримку текстової векторизації та можливість створення embedding-рівня.

Таблиця 1.5 – Порівняння бібліотек та фреймворків для реалізації класифікації текстів

<i>Інструмент</i>	<i>Тип рішення</i>	<i>Сильні сторони</i>	<i>Практичне обмеження</i>
scikit-learn	Бібліотека класичного машинного навчання	TF-IDF, класифікатори, pipeline, метрики	Обмежена підтримка глибоких нейронних мереж
spracy	NLP-бібліотека	Токенізація, лінгвістична обробка, категоризація документів	Менша гнучкість для побудови складних нейромереж
TensorFlow/Keras	Фреймворк глибокого навчання	Зручна побудова нейромереж, embedding-шари, готові механізми навчання	Потреба в налаштуванні архітектури
PyTorch	Фреймворк глибокого навчання	Гнучке керування навчальним циклом, зручність досліджень	Вища складність коду для початківця
Hugging Face Transformers	Бібліотека трансформерних моделей	Попередньо навчені моделі, донавчання, сучасні архітектури	Підвищені вимоги до ресурсів

Хмарні сервіси мають перевагу в швидкому підключенні готових можливостей, однак вони не дають повного контролю над внутрішніми етапами навчання. Контроль над моделлю має істотне значення, оскільки необхідно описати набір даних, попередню обробку, архітектуру, навчання, тестування та оцінку результатів. Тому Google Cloud Natural Language, Amazon Comprehend і Azure AI Language [21] варто розглядати як аналоги, а не як основу власної реалізації.

В підсумку для подальшої розробки програмної системи найдоцільніше обрати Python як мову реалізації, scikit-learn як засіб побудови базової порівняльної моделі, TensorFlow/Keras як основу нейромережевої реалізації, pandas і NumPy як інструменти роботи з набором текстових записів, а matplotlib

або вбудовані засоби Python як допоміжні інструменти візуалізації результатів. Hugging Face Transformers можна залишити як перспективний варіант для порівняння з попередньо навченими моделями, однак основна реалізація має бути побудована на більш контрольованій та зрозумілій архітектурі [22].

1.5 Постановка задачі дослідження з визначенням вхідних даних та критеріїв оцінювання результатів

Після аналізу сучасних методів тематичної класифікації текстів, алгоритмів машинного навчання, нейромережових архітектур і готових програмних засобів постає потреба в формалізованій постановці задачі дослідження. Практичне значення поставленої задачі полягає в створенні рішення, яке може приймати текст на вхід, виконувати його попередню обробку, перетворювати мовний матеріал на числове представлення, здійснювати класифікацію та надавати результат у вигляді прогнозованої категорії.

Задача дослідження полягає в побудові нейромережової моделі класифікації текстів, яка навчається на прикладах з відомими мітками класів і надалі використовується для розподілу нових текстових записів за наперед визначеними категоріями. В процесі виконання роботи необхідно не лише реалізувати алгоритм навчання, але й забезпечити повний технологічний цикл роботи з текстом, а саме, завантаження датасету, очищення записів, токенизацію, формування словника, перетворення текстів на числові послідовності, навчання моделі, перевірку якості, збереження результатів та демонстрацію роботи через програмний інтерфейс [23].

Об'єктом програмного опрацювання є датасет текстових записів, в якому кожен приклад має змістову частину та відповідну мітку категорії. Дана структура набору дозволяє застосувати контрольоване навчання, оскільки

модель отримує не лише текст, але й правильну відповідь, з якою порівнюється прогноз під час навчання. Категорії можуть відповідати тематичним рубрикам новин, типам звернень користувачів, класам відгуків, видам повідомлень або іншим групам, визначеним структурою обраного датасету.

Вхідний датасет має бути підготовлений у форматі, придатному для автоматичного зчитування, наприклад у вигляді CSV-файлу або іншої табличної структури, де окремі поля відповідають тексту та категорії. Перед навчанням необхідно перевірити наявність порожніх записів, дублювання, зайвих службових символів та надмірно коротких фрагментів, оскільки низька якість датасету знижує надійність підсумкової моделі [24].

Попередня обробка текстів має забезпечити уніфіковане подання мовного матеріалу. До неї належать приведення символів до єдиного регістру, видалення зайвих пробілів, видалення службових елементів, токенизація та перетворення слів або токенів на числові індекси. Для нейромережевої моделі важливим етапом є формування послідовностей однакової довжини, оскільки вхідний шар мережі потребує узгодженого формату. Коротші послідовності доповнюються службовими значеннями, а надмірно довгі тексти обрізаються до встановленої максимальної довжини.

Вихідним результатом роботи моделі є прогнозована категорія тексту. В багатокласовій класифікації, нейромережа формує набір імовірнісних оцінок для всіх класів, після чого обирається категорія з найбільшим значенням. Якщо модель має визначати лише одну тему для кожного тексту, потрібно використовувати вихідний шар з функцією softmax. Якщо програмна система надалі буде розширюватися до багатоміткової класифікації, де один текст може належати одразу до кількох категорій, архітектуру потрібно змінювати [24]. Вхідні та вихідні параметри моделі класифікації текстів наведено в таблиці 1.6.

Таблиця 1.6 – Вхідні та вихідні параметри моделі класифікації текстів

<i>Параметр</i>	<i>Тип параметра</i>	<i>Очікуване подання</i>	<i>Призначення</i>
Вхідний текст	Текстовий	Рядок природною мовою	Початковий матеріал для аналізу
Категорія тексту	Категоріальний	Назва класу або числовий код	Еталонне значення під час навчання
Токенізований текст	Проміжний	Послідовність токенів	Підготовка до числового кодування
Числова послідовність	Проміжний	Масив індексів однакової довжини	Вхід для нейронної мережі
Прогноз моделі	Вихідний	Вектор імовірностей за класами	Основа для вибору категорії
Підсумкова мітка	Вихідний	Назва прогнозованої категорії	Результат роботи системи для користувача

Критеріями оцінювання результатів мають бути кількісні показники, які дозволяють порівняти якість побудованої моделі з базовими алгоритмами. Загальна точність показує частку правильних прогнозів, проте для повноцінної оцінки її недостатньо. Для кожного класу потрібно аналізувати precision, recall і F1-score, оскільки вони показують не лише загальну кількість правильних відповідей, а й характер помилок. Додатково необхідно використовувати матрицю помилок, яка дає змогу визначити, між якими категоріями виникає найбільше неправильних прогнозів [25].

Під час дослідження потрібно порівнювати нейромережеву модель з базовим алгоритмом, побудованим на TF-IDF-представленні та класичному класифікаторі. Дане порівняння потрібне для обґрунтування практичної цінності нейромережевого рішення. Якщо нейронна мережа демонструє вищий F1-score або кращу рівномірність розпізнавання класів, можна зробити висновок про доцільність її застосування для вибраного корпусу. Якщо різниця між моделями незначна, аналіз має враховувати додаткові фактори, зокрема швидкість навчання, простоту запуску, вимоги до ресурсів і зручність подальшого розширення.

Практичний результат дослідження має полягати у створенні програмної системи, яка забезпечує завантаження датасету, підготовку текстів, навчання моделі, оцінювання результатів та класифікацію нового текстового запису. Інтерфейс користувача повинен дозволити вводити або завантажувати текст, запускати класифікацію та переглядати отриману категорію [25].

Очікувана якість програмної реалізації визначається не лише високими значеннями метрик, а й стабільністю роботи на тестових даних. Модель повинна коректно опрацьовувати тексти різної довжини, не переривати виконання через некоректні символи, давати результат в зрозумілому для користувача вигляді та зберігати придатність до повторного навчання після оновлення корпусу. Важливою вимогою є відтворюваність експерименту, тому під час реалізації потрібно зафіксувати етапи підготовки даних, параметри навчання та умови оцінювання.

Висновки до розділу 1

В першому розділі розглянуто актуальність автоматичної класифікації текстової інформації та визначено її практичне значення для інформаційних систем, медіаресурсів, клієнтської підтримки, освіти, державного управління та аналітичної діяльності. Встановлено, що ручне опрацювання великих обсягів текстів є недостатньо ефективним, а автоматизований розподіл повідомлень за категоріями дозволяє підвищити швидкість обробки та забезпечити єдині правила прийняття рішень.

Проаналізовано сучасні підходи до розпізнавання тематичної належності текстових даних, серед яких розглянуто правилі методи, статистичне машинне навчання, TF-IDF-представлення, векторні подання, нейронні мережі та трансформерні архітектури. Класичні алгоритми залишаються важливими для побудови базового рівня порівняння, а нейромережеві моделі краще

відповідають задачам, де потрібно працювати з прихованими закономірностями текстового матеріалу.

Якість моделі потрібно оцінювати не лише за загальною точністю, а й за precision, recall, F1-score, матрицею помилок, швидкістю навчання та придатністю до практичного використання.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТІВ

2.1 Формалізація задачі автоматичного віднесення текстів до заданих категорій

Автоматичне віднесення текстів до заданих категорій розглядається як задача контрольованого машинного навчання, де модель навчається на прикладах з відомими мітками класів і надалі виконує прогноз для нових текстових записів. На вхід програмної системи подається текст природною мовою, а на виході формується категорія, що найкраще відповідає змісту вхідного повідомлення. Формалізація потрібна для чіткого визначення структури даних, логіки перетворення тексту, вимог до моделі та критеріїв оцінювання результату.

В загальному вигляді датасет текстів можна подати як множину пар:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\},$$

де D – розмічений набір текстових даних, x_i – окремий текстовий запис, y_i – мітка класу, яка відповідає запису, n – кількість прикладів в датасеті. Кожна мітка належить до наперед визначеної множини категорій:

$$Y = \{c_1, c_2, \dots, c_k\},$$

де Y – множина можливих класів, c_j – окрема категорія, k – кількість категорій. Завдання моделі полягає в побудові функції:

$$f: X \rightarrow Y,$$

де X – множина текстових записів, а $f(x)$ — прогнозована категорія для вхідного тексту. Після навчання, функція повинна для невідомого запису x_{new} визначити клас y_{pred} , який з найбільшою ймовірністю відповідає змісту тексту [26].

Текст природною мовою не може безпосередньо подаватись на вхід нейронної мережі, оскільки обчислювальна модель працює з числовими значеннями. Формалізація задачі охоплює не лише визначення вхідних та вихідних множин, але й опис проміжного перетворення тексту. Вхідний запис спочатку проходить очищення, після чого розбивається на токени. Далі кожен токен отримує числовий індекс відповідно до словника, сформованого на основі навчального датасету. Після індексації, текст перетворюється на послідовність чисел фіксованої довжини, придатну для подання на вхід нейромережевої моделі.

Процес перетворення тексту можна подати як послідовність відображень:

$$x_i \rightarrow t_i \rightarrow s_i \rightarrow v_i,$$

де x_i – початковий текст, t_i – набір токенів після токенизації, s_i – числова послідовність індексів, v_i – векторне або матричне представлення тексту, яке використовується нейронною мережею. При використанні embedding-шару кожен індекс токена перетворюється на щільний вектор, а весь текст подається як матриця векторних представлень [26]. Перетворення текстового запису перед поданням до моделі наведено в таблиці 2.1.

Таблиця 2.1 – Перетворення текстового запису перед поданням до моделі

<i>Етап перетворення</i>	<i>Вхід етапу</i>	<i>Вихід етапу</i>	<i>Призначення</i>
Очищення тексту	Початковий рядок	Нормалізований текст	Усунення зайвих символів, пробілів і технічного шуму
Токенізація	Нормалізований текст	Послідовність токенів	Поділ тексту на окремі мовні одиниці
Індексація	Послідовність токенів	Послідовність числових кодів	Перехід від слів до числового подання
Вирівнювання довжини	Числова послідовність	Послідовність однакової довжини	Узгодження формату вхідних даних
Векторизація	Індекси токенів	Векторне представлення	Формування ознак для нейронної мережі
Класифікація	Векторне представлення	Імовірності класів	Визначення належності тексту до категорії

У багатокласовій постановці кожен текст належить лише до однієї категорії. Для прогнозування модель формує вектор імовірностей:

$$y_1 = \{p_1, p_2, \dots, p_k\},$$

де p_j – оцінка ймовірності належності тексту до класу c_j . Значення вектора мають відповідати умові:

$$\sum_{j=1}^k p_j = 1$$

Підсумкова категорія визначається за максимальним значенням у векторі прогнозу:

$$y_{pred} = \arg \max_{c_j \in Y} p_j$$

Ця формула означає, що модель обирає клас з найбільшою прогнозованою ймовірністю. Наприклад, якщо система працює з категоріями «спорт», «економіка», «технології» та «культура», то після опрацювання тексту вона повертає набір ймовірностей для всіх категорій, а користувач бачить назву класу з найбільшим значенням [27].

Формалізація задачі також передбачає визначення структури навчальної, валідаційної та тестової вибірок. Навчальна частина використовується для налаштування параметрів моделі. Валідаційна частина потрібна для проміжного контролю якості під час навчання та вибору параметрів архітектури. Тестова частина не бере участі в навчанні та застосовується для підсумкової оцінки здатності моделі працювати з новими текстами. Поділ датасету на частини має виконуватись до навчання, щоб уникнути потрапляння однакових або надто близьких записів до різних вибірок.

Для нейромережевої моделі важливим є визначення розмірностей вхідних і вихідних даних. Якщо максимальна довжина текстової послідовності дорівнює L , то кожен текст після попередньої обробки подається як вектор довжини L . Якщо розмір embedding-вектора дорівнює d , то після embedding-шару текст має вигляд матриці розміром $L \times d$. Вихідний шар моделі має містити k нейронів, де k відповідає кількості категорій. Для багатокласової класифікації на виході доцільно застосувати функцію softmax, а як функцію втрат – categorical cross-entropy або sparse categorical cross-entropy залежно від формату міток [27].

Під час навчання модель мінімізує функцію втрат, яка відображає різницю між прогнозованим та правильним класом. Для одного прикладу в багатокласовій класифікації функція втрат може бути подана у вигляді:

$$L(y, \tilde{y}) = - \sum_{j=1}^k y_j \log(\tilde{y}_j)$$

де y_j – правильне значення для класу j , а $\tilde{y}_j$ – прогнозована ймовірність для класу j . Якщо модель правильно визначає категорію з високою впевненістю, значення втрат зменшується. Якщо прогноз відрізняється від правильного класу, значення втрат зростає, що змушує алгоритм навчання коригувати ваги мережі.

Формалізована задача має враховувати й можливі обмеження. Текстові записи можуть мати різну довжину, містити помилки, скорочення, іноземні слова, службові символи або фрагменти без змістового навантаження. Категорії можуть бути представлені нерівномірно, що впливає на якість класифікації менш чисельних класів. Крім того, окремі теми можуть мати спільну лексику, через що модель здатна плутати близькі за змістом категорії. Для зменшення впливу перелічених факторів необхідно виконати якісну попередню обробку, контролювати розподіл класів та використовувати кілька метрик оцінювання [27].

Результат роботи класифікатора повинен оцінюватися на основі метрик, які відображають різні аспекти якості. Ассурасу показує загальну частку правильних відповідей, але не дає повного уявлення про поведінку моделі за окремими категоріями. Precision характеризує надійність прогнозів для класу, recall показує здатність знаходити всі записи відповідної категорії, а F1-score поєднує обидва показники. Матриця помилок потрібна для аналізу напрямів помилкової класифікації. Якщо модель часто плутає певні класи, необхідно переглянути датасет, ознаки або архітектуру.

В рамках програмної реалізації, формалізована задача має бути перетворена на алгоритмічну послідовність роботи системи. Користувач або дослідник завантажує датасет текстів, програма виконує перевірку структури даних, здійснює попередню обробку, формує словник, навчає модель, обчислює метрики та зберігає результати. Після завершення навчання система повинна приймати новий текст та повертати прогнозовану категорію. Важливо, щоб всі етапи були відтворюваними, а параметри навчання фіксувались для подальшого аналізу [27].

Формалізація задачі автоматичного віднесення текстів до заданих категорій створює методичну основу для побудови програмної системи. Вона визначає структуру датасету, математичну постановку класифікації, вимоги до перетворення текстів, параметри нейромережевої моделі та критерії перевірки результатів. Надалі описана постановка дає змогу обґрунтовано перейти до вибору методів обробки текстової інформації, побудови архітектури моделі та реалізації програмного продукту для класифікації текстів.

2.2 Вибір методів та алгоритмів обробки текстової інформації

Розробка моделі класифікації текстів потребує не лише вибору нейромережевої архітектури, а й формування коректної схеми підготовки датасету до контрольованого навчання. На відміну від задач, орієнтованих на скорочення або переказ тексту, класифікація передбачає встановлення відповідності між текстовим записом і наперед визначеною категорією. Основну увагу необхідно приділити не відбору ключових речень, а забезпеченню правильної структури навчальних прикладів, кодуванню міток, збереженню балансу категорій, перетворенню тексту на числові послідовності та формуванню умов для стабільного навчання моделі.

В програмній системі текстовий датасет розглядається як набір записів, де кожен приклад містить текст та відповідну категорію. Формально розмічений датасет текстових даних можна подати у вигляді множини пар:

$$D = \{(x_i, y_i)\}_{i=1}^N$$

де D – розмічений датасет текстових даних; x_i – окремий текстовий запис; y_i – мітка категорії, що відповідає тексту x_i ; N – загальна кількість записів в датасеті.

Множина доступних категорій визначається як:

$$C = \{c_1, c_2, \dots, c_K\},$$

де C – множина категорій; c_j – окрема тематична категорія; K – кількість класів в задачі класифікації.

Якість класифікації залежить від того, наскільки коректно підготовлено обидві складові. Текстова частина повинна бути перетворена на числове представлення, придатне для нейронної мережі, а мітки класів мають бути закодовані у форматі, сумісному з функцією втрат. Необхідно контролювати розподіл категорій, оскільки нерівномірність датасету може призвести до переважного навчання моделі на найбільш представлених класах [28]. Методи обробки текстових даних, орієнтовані на задачу класифікації наведено в таблиці 2.2.

Початковим етапом є перевірка структури датасету. Для задачі класифікації недостатньо мати лише набір текстів, оскільки модель навчається на зв'язку між вхідним повідомленням та правильною відповіддю. Тому кожен рядок повинен містити текстове поле та поле категорії. Якщо запис не має тексту, містить порожню мітку або дублює інший приклад без потреби, він може погіршити навчання. Перевірка структури дозволяє сформувати коректний набір прикладів, в якому кожен текст має однозначну належність до одного класу [28].

Особливе значення має кодування міток класів. У вхідному файлі категорії можуть бути подані текстовими назвами, наприклад «спорт», «економіка», «технології», «культура». Нейромережева модель не працює з назвами категорій безпосередньо, тому кожна мітка перетворюється на числовий код.

Таблиця 2.2 – Методи обробки текстових даних, орієнтовані на задачу класифікації

<i>Метод</i>	<i>Призначення</i>	<i>Значення для класифікаційної моделі</i>
Перевірка структури датасету	Контроль наявності тексту та категорії в кожному записі	Запобігає навчанню на неповних або некоректних прикладах
Кодування міток класів	Перетворення назв категорій на числові значення	Забезпечує сумісність вихідних даних з функцією втрат
Стратифікований поділ	Розподіл датасету зі збереженням пропорцій класів	Дає змогу отримати коректну навчальну, валідаційну та тестову вибірки
Аналіз дисбалансу класів	Виявлення надмірної або недостатньої представленості категорій	Допомагає обрати засоби компенсації нерівномірності
Підслівна токенизація	Поділ тексту на слова або частини слів	Покращує роботу з рідкісними словами й новими словоформами
Обмеження словника	Відбір найінформативніших токенів за частотою	Зменшує розмірність простору ознак
Padding і truncation	Вирівнювання послідовностей до єдиної довжини	Формує однаковий формат входу для нейромережі
Ймовірнісна інтерпретація виходу	Перетворення результату softmax на категорію	Дає змогу показати не лише клас, але й рівень упевненості

Кодування текстових назв категорій можна подати через відображення:

$$g(c_j) = j, j = 0, 1, \dots, K - 1,$$

де $g(c_j)$ – числовий код категорії c_j ; j – порядковий номер класу; K – загальна кількість категорій.

Для кожного текстового запису формується числова мітка:

$$\tilde{y}_i = g(y_i)$$

де $\tilde{y}_i$ – числове представлення категорії для запису x_i ; y_i – початкова текстова мітка; $g(y_i)$ – результат перетворення назви категорії на числовий код.

За потреби використання one-hot-подання мітка класу записується у вигляді вектора:

$$y_{ij} = \begin{cases} 1, y_i = c_j \\ 0, y_i \neq c_j \end{cases}$$

де y_{ij} – елемент вектора мітки для прикладу i та категорії j . Значення 1 відповідає правильному класу, а значення 0 встановлюється для інших категорій.

Якщо використовується функція втрат sparse categorical cross-entropy, достатньо подати мітки як цілі числа. Якщо застосовується categorical cross-entropy, категорії додатково перетворюються на one-hot-подання, де правильному класу відповідає одиниця, а іншим класам, нулі. Варіанти кодування категорій у задачі класифікації текстів наведено в таблиці 2.3.

Таблиця 2.3 – Варіанти кодування категорій в задачі класифікації текстів

<i>Спосіб кодування</i>	<i>Приклад подання</i>	<i>Доцільність використання</i>
Числове кодування	«спорт» → 0, «економіка» → 1, «технології» → 2	Зручне для sparse categorical cross-entropy
One-hot-кодування	«технології» → [0, 0, 1, 0]	Зручне для categorical cross-entropy
Зворотне відображення	2 → «технології»	Потрібне для виведення зрозумілого результату користувачеві
Таблиця відповідності класів	Код, назва категорії, кількість прикладів	Потрібна для контролю структури корпусу й аналізу результатів

Після кодування категорій необхідно виконати аналіз розподілу класів. Кількість прикладів, що належать до окремої категорії, визначається за формулою:

$$n_j = \sum_{i=1}^N I(y_i = c_j)$$

де n_j – кількість текстових записів категорії c_j ; N – загальна кількість записів в датасеті; $I(y_i = c_j)$ – індикаторна функція, яка набуває значення 1, якщо мітка y_i відповідає категорії c_j , і значення 0 в іншому випадку.

Частка категорії в датасеті обчислюється за формулою:

$$r_j = \frac{n_j}{N}$$

де r_j – частка категорії c_j в датасеті; n_j – кількість записів відповідної категорії; N – загальний обсяг датасету.

Значення r_j дозволяє оцінити рівномірність представлення категорій.

Якщо одна категорія містить значно більше прикладів, ніж інші, модель може навчитись переважно прогнозувати найбільший клас. В результаті, загальна точність виглядатиме прийнятною, але якість для менш представлених категорій буде низькою. Для запобігання такій ситуації потрібно визначити кількість прикладів в кожній категорії ще до поділу датасету. Якщо дисбаланс суттєвий, до навчання можна додати ваги класів, які підсилюють вплив рідкісних категорій на функцію втрат [28].

Стратифікований поділ датасету є важливим методом, безпосередньо пов'язаним з класифікацією. Його зміст полягає в тому, що навчальна, валідаційна та тестова вибірки зберігають приблизно однакові пропорції класів. Без стратифікації може виникнути ситуація, коли певна категорія майже не потрапить до тестової частини або буде слабо представлена під час навчання. Відповідно, результати оцінювання стануть ненадійними. Стратифікований

поділ підвищує об'єктивність експерименту, оскільки модель перевіряється на наборі, що відображає реальну структуру корпусу.

Стратифікований поділ датасету передбачає збереження пропорцій категорій в навчальній, валідаційній та тестовій вибірках. Якщо α, β, γ позначають частки навчальної, валідаційної та тестової вибірок, то виконується умова:

$$\alpha + \beta + \gamma = 1,$$

Для кожної категорії кількість прикладів у відповідних вибірках можна подати так:

$$\begin{aligned} n_j^{train} &\approx \alpha n_j, \\ n_j^{val} &\approx \beta n_j, \\ n_j^{test} &\approx \gamma n_j, \end{aligned}$$

де n_j^{train} – кількість записів категорії c_j в навчальній вибірці; n_j^{val} – кількість записів категорії c_j у валідаційній вибірці; n_j^{test} – кількість записів категорії c_j в тестовій вибірці; n_j – загальна кількість записів категорії c_j в датасеті.

Завдяки такому поділу кожна частина датасету зберігає наближене співвідношення категорій. Це підвищує надійність експерименту, оскільки модель навчається та перевіряється на вибірках з подібною структурою класів.

Для представлення тексту в нейромережевій моделі обирається токенизація з можливістю обробки невідомих слів. В звичайному словесному варіанті кожне слово отримує власний індекс. Проблема виникає тоді, коли користувач вводить слово, відсутнє в навчальному словнику. Для обробки таких випадків потрібно використовувати службовий токен невідомої одиниці. Більш гнучким варіантом є підслівна токенизація, за якої рідкісне або нове слово може бути розкладене на менші фрагменти.

Обмеження словника необхідне для зменшення розмірності моделі. Якщо включити до словника всі слова датасету, embedding-шар може стати надмірно великим, а рідкісні слова не дадуть істотного приросту якості [28]. Рациональніше залишити найбільш вживані токени, які формують основну змістову структуру категорій. Інші слова передаються через службовий індекс невідомого токена. Параметри подання тексту для нейромережевої класифікації наведено в таблиці 2.4.

Таблиця 2.4 – Параметри подання тексту для нейромережевої класифікації

<i>Параметр</i>	<i>Призначення</i>	<i>Вплив на модель</i>
Розмір словника	Визначає кількість токенів, доступних для індексації	Впливає на розмір embedding-шару
Службовий токен невідомого слова	Забезпечує обробку нових або рідкісних слів	Підвищує стійкість прогнозування
Максимальна довжина послідовності	Встановлює кількість токенів на вході моделі	Впливає на обсяг збереженої інформації
Padding	Доповнює короткі тексти службовими значеннями	Забезпечує однакову довжину вхідних масивів
Truncation	Скорочує надмірно довгі тексти	Запобігає перевищенню встановленого формату входу
Розмір embedding-вектора	Визначає кількість ознак для одного токена	Впливає на здатність моделі виявляти закономірності

Вирівнювання довжини послідовностей є обов'язковим для подання даних до нейронної мережі. Тексти датасету мають різну кількість слів, але модель потребує однакового формату входу. Для цього короткі записи доповнюються службовими нульовими значеннями, а надмірно довгі обмежуються максимальною довжиною. Значення максимальної довжини не варто обирати випадково. Воно має визначатись після аналізу розподілу довжин текстів, щоб більшість записів зберігала змістову інформацію, а вхідний масив не ставав надмірно великим.

Для підвищення стійкості навчання необхідно використати dropout і ваги класів. Dropout випадково вимикає частину нейронів під час навчання, що зменшує ризик запам'ятовування навчальних прикладів. Ваги класів потрібні тоді, коли датасет має нерівномірний розподіл категорій. Якщо рідкісний клас отримує більшу вагу, помилки за ним сильніше впливають на функцію втрат. Завдяки цьому, модель отримує стимул краще розпізнавати менш представлені категорії, а не лише найбільший клас [28].

Нейромережева модель з вихідним шаром softmax повертає не тільки назву прогнозованого класу, але й розподіл імовірностей за всіма категоріями. Методи стабілізації навчання та підвищення надійності прогнозу наведено в таблиці 2.5.

Таблиця 2.5 – Методи стабілізації навчання та підвищення надійності прогнозу

<i>Метод</i>	<i>Проблема, яку він зменшує</i>	<i>Практичний ефект</i>
Dropout	Перенавчання на навчальній вибірці	Підвищення здатності до узагальнення
Ваги класів	Ігнорування менш представлених категорій	Покращення якості для рідкісних класів
Early stopping	Надмірне навчання після погіршення валідаційної якості	Збереження моделі з кращим узагальненням
Фіксація random seed	Нестабільність результатів між запусками	Відтворюваність експерименту
Поріг впевненості	Невпевнене віднесення тексту до категорії	Обережніше подання результату користувачеві
Збереження історії навчання	Відсутність даних для аналізу поведінки моделі	Можливість побудови графіків accuracy і loss

Алгоритмічна логіка підготовки даних у розроблюваній системі має відрізнятись від звичайного опису очищення текстів. Спочатку програма перевіряє наявність двох обов'язкових полів, а саме, тексту та категорії. Далі виконується аналіз кількості прикладів в кожному класі, після чого формується

таблиця відповідності між назвами категорій та числовими кодами. Потім датасет поділяється на навчальну, валідаційну та тестову частини з використанням стратифікації. Токенізатор навчається лише на навчальній частині, щоб інформація з тестової вибірки не впливала на підготовку словника. Після цього тексти всіх вибірок перетворюються на числові послідовності однакової довжини [28].

Запропонований набір методів створює основу для подальшої побудови нейромережевої моделі. Він дозволяє підготувати дані у відтворюваному форматі, зменшити вплив дисбалансу класів, забезпечити однакову структуру входу та отримати результат у вигляді зрозумілої категорії з оцінкою впевненості.

2.3 Побудова нейромережевої моделі для класифікації текстових даних та опис метрик оцінювання її ефективності

Побудова нейромережевої моделі для класифікації текстових даних ґрунтується на перетворенні текстового запису на числову послідовність і подальшому визначенні категорії, до якої належить вхідний матеріал. На відміну від генеративних задач обробки природної мови, де модель створює новий текст, класифікаційна модель не формує мовний виклад. Її результатом є прогнозована мітка класу та набір імовірнісних оцінок, які показують рівень належності тексту до кожної категорії.

В розроблюваній системі нейромережева модель має працювати з датасетом, де кожен текстовий запис пов'язаний з однією категорією. Після попередньої підготовки, текст перетворюється на послідовність числових індексів, а категорія, на числову мітку. Далі індекси надходять до embedding-шару, який формує щільні векторні представлення токенів. На наступних рівнях, модель узагальнює отримані ознаки та передає їх до класифікаційного

блоку. Вихідний шар з функцією softmax формує розподіл імовірностей за всіма класами. Основні компоненти нейромережевої моделі класифікації текстів наведено в таблиці 2.6.

Таблиця 2.6 – Основні компоненти нейромережевої моделі класифікації текстів

<i>Компонент моделі</i>	<i>Вхідні дані</i>	<i>Вихідні дані</i>	<i>Призначення</i>
Вхідний шар	Послідовність індексів токенів	Масив фіксованої довжини	Приймає підготовлений текстовий запис
Embedding-шар	Індекси токенів	Матриця щільних векторів	Формує числове подання мовних одиниць
Шар узагальнення ознак	Матриця векторів	Компактний вектор тексту	Виділяє змістові ознаки для класифікації
Повнозв'язний шар	Узагальнений вектор	Набір прихованих ознак	Встановлює залежності між ознаками та класами
Dropout-шар	Приховані ознаки	Проріджене представлення	Зменшує ризик перенавчання
Вихідний softmax-шар	Приховані ознаки	Вектор імовірностей	Визначає належність тексту до категорій

Embedding-шар є одним з ключових елементів моделі, оскільки він замінює прості числові коди токенів на векторні представлення. Кожен токен отримує вектор фіксованої довжини, параметри якого змінюються під час навчання. Якщо певні слова часто з'являються в текстах однієї тематики, модель поступово враховує їх значення для відповідної категорії. На відміну від звичайного частотного подання, embedding-рівень дає змогу отримати компактний простір ознак, який краще пристосовується до конкретного датасету [29].

Після embedding-шару потрібно перетворити послідовність векторів на представлення всього тексту. Глобальне усереднення зменшує кількість параметрів, прискорює навчання та добре підходить для першої робочої версії програми. Згортковий блок може виділяти локальні словосполучення,

характерні для певних категорій. Вибір між зазначеними варіантами залежить від обсягу датасету, довжини текстів та доступних обчислювальних ресурсів. Архітектура нейромережевої моделі багатокласової класифікації текстів наведено на рисунку 2.1.

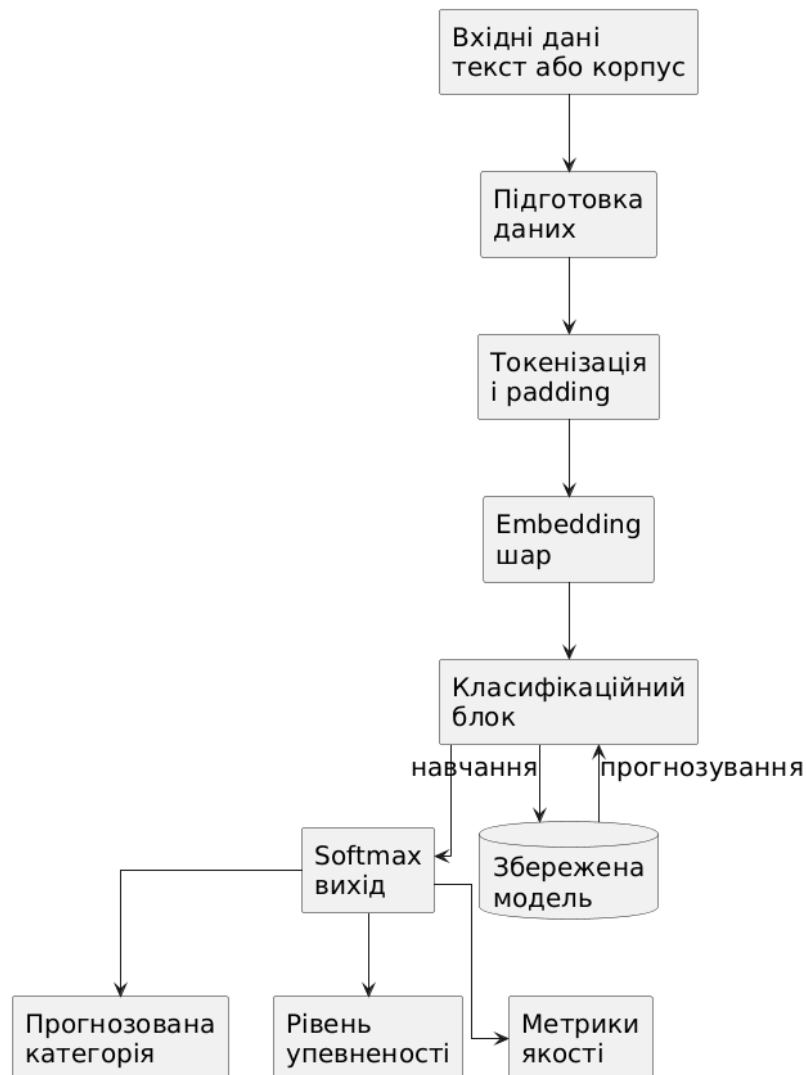


Рисунок 2.1 – Архітектура нейромережевої моделі багатокласової класифікації текстів

Після проходження всіх шарів модель повертає вектор імовірностей, в якому кожне значення відповідає окремому класу. Категорія з найбільшим значенням приймається як прогноз моделі [29].

Формально підготовлений текст можна подати у вигляді послідовності:

$$x = [w_1, w_2, \dots, w_L],$$

де w_i є індексом токена, а L позначає встановлену довжину вхідної послідовності. Embedding-шар виконує перехід від індексу токена до вектора:

$$E(w_i) \in R^d,$$

де d є розмірністю embedding-вектора. Після перетворення весь текст набуває вигляду матриці:

$$X_E \in R^{L \times d}.$$

Вихідний softmax-шар є важливим для багатокласової класифікації, оскільки він перетворює результати останнього шару на розподіл імовірностей. Сума всіх значень у векторі дорівнює одиниці, тому результат можна інтерпретувати як рівень впевненості моделі щодо кожної категорії. В програмному інтерфейсі необхідно виводити не лише назву прогнозованого класу, але й значення найбільшої ймовірності. Якщо впевненість низька, система може позначати результат як невизначений або такий, що потребує додаткової перевірки користувачем. Параметри нейромережевої моделі класифікації наведено в таблиці 2.7.

При нерівномірному розподілі категорій в датасеті варто використати ваги класів. Якщо певна категорія має невелику кількість прикладів, помилки за нею можуть майже не впливати на загальне значення втрат. Вагові коефіцієнти дають змогу підсилити роль менш представлених класів під час навчання. В результаті модель отримує стимул краще розпізнавати не лише найбільші категорії, але й класи з меншою кількістю прикладів [29].

Таблиця 2.7 – Параметри нейромережевої моделі класифікації

<i>Параметр</i>	<i>Зміст</i>	<i>Вплив на результат</i>
Розмір словника	Кількість токенів, доступних для індексації	Визначає обсяг embedding-шару
Максимальна довжина послідовності	Кількість токенів, які подаються на вхід	Впливає на обсяг збереженої інформації
Розмір embedding-вектора	Кількість ознак для одного токена	Визначає місткість векторного подання
Кількість прихованих нейронів	Обсяг внутрішнього класифікаційного блоку	Впливає на здатність розділяти категорії
Dropout	Частка випадково вимкнених нейронів під час навчання	Зменшує ризик перенавчання
Кількість класів	Число категорій корпусу	Визначає розмір вихідного softmax-шару
Розмір пакета	Кількість прикладів за один крок навчання	Впливає на швидкість та стабільність оптимізації
Кількість епох	Кількість проходжень навчальної вибірки	Визначає тривалість налаштування ваг

Для контролю перенавчання застосовуються dropout, валідаційна вибірка та рання зупинка. Dropout зменшує залежність моделі від окремих нейронів, валідаційна вибірка показує якість на даних, які не використовуються для оновлення ваг, а рання зупинка дозволяє завершити навчання після погіршення валідаційного результату. В поєднанні зазначені засоби підвищують здатність моделі працювати з новими текстами, а не лише з прикладами навчального корпусу. Засоби підвищення стійкості нейромережевої моделі наведена в таблиці 2.8.

Таблиця 2.8 – Засоби підвищення стійкості нейромережевої моделі

<i>Засіб</i>	<i>Призначення</i>	<i>Очікуваний ефект</i>
Dropout	Обмеження надмірного запам'ятовування навчальних прикладів	Краще узагальнення на нових текстах
Class weights	Компенсація нерівномірного розподілу категорій	Покращення якості для менш представлених класів
Early stopping	Зупинка навчання після погіршення валідаційної якості	Захист від перенавчання
Validation split	Перевірка якості під час навчання	Контроль стабільності моделі
Фіксація random seed	Повторюваність експерименту	Можливість перевірити результати повторним запуском
Збереження найкращої моделі	Фіксація варіанта з найкращим валідаційним показником	Запобігання втраті оптимального стану ваг

Оцінювання нейромережевої моделі має виконуватись на тестовій вибірці, яка не брала участі в навчанні. Під час перевірки модель формує прогноз для кожного тексту, після чого прогнозована мітка порівнюється з правильною категорією. Для багатокласової класифікації недостатньо обмежуватись лише загальною точністю, оскільки вона може приховувати слабе розпізнавання окремих категорій. Тому потрібно використовувати ассурасу, precision, recall, F1-score і матрицю помилок [29].

В багатокласовій класифікації показники precision, recall і F1-score необхідно розраховувати для кожної категорії окремо. Після цього можна визначити macro average та weighted average. Macro average показує середню якість для всіх категорій без урахування їх чисельності. Weighted average враховує кількість прикладів в кожному класі. Якщо macro average помітно нижчий за weighted average, модель може добре працювати з великими класами, але слабше розпізнавати рідкісні категорії.

Матриця помилок має особливе значення для аналізу результатів. Вона показує, які категорії модель розпізнає правильно, а між якими виникає плутанина. Наприклад, тексти про науку і технології можуть мати спільну

лексику, через що модель здатна відносити частину записів до сусідньої тематики. Аналіз матриці помилок дозволяє виявити проблемні класи та визначити, чи потрібно збільшити кількість прикладів, змінити параметри моделі або переглянути правила підготовки датасету [29]. Інтерпретація результатів оцінювання моделі наведена в таблиці 2.9.

Таблиця 2.9 – Інтерпретація результатів оцінювання моделі

<i>Результат оцінювання</i>	<i>Можлива причина</i>	<i>Напрямок покращення</i>
Висока accuracy і низький F1-score для окремого класу	Нерівномірний розподіл категорій	Використання class weights або доповнення корпусу
Низький precision для категорії	Модель надмірно часто обирає відповідний клас	Перегляд навчальних прикладів і меж між класами
Низький recall для категорії	Модель пропускає частину текстів класу	Збільшення кількості прикладів або зміна параметрів
Зростання training accuracy і падіння validation accuracy	Перенавчання	Dropout, early stopping або спрощення архітектури
Високий validation loss	Недостатня здатність до узагальнення	Перегляд архітектури й режиму навчання
Часті помилки між двома класами	Тематична близькість категорій	Аналіз корпусу та уточнення міток

Для практичної системи важливо зберігати не лише навчену модель, але й службові об'єкти, без яких прогнозування нових текстів буде некоректним. До них належать токенизатор, таблиця відповідності між числовими кодами та назвами категорій, параметри максимальної довжини послідовності та звіт з метриками. Під час прогнозування нового тексту програма повинна застосовувати саме той словник і ті параметри, які використовувалися під час навчання. Повторне створення словника для окремого введеного тексту призведе до неправильного індексування.

Побудована нейромережева модель має забезпечувати повний класифікаційний цикл: отримання підготовленого текстового запису,

перетворення його на числову послідовність, формування embedding-представлення, визначення ймовірностей класів і повернення прогнозованої категорії. Її ефективність оцінюється не одним показником, а сукупністю метрик, які відображають загальну якість, надійність прогнозів, повноту розпізнавання категорій і характер помилок.

2.4 Розробка та проєктування архітектури програми та опис основних етапів та логіки роботи

Проєктування програмної системи для класифікації текстів має спиратися на особливості задачі контрольованого навчання, де кожен текстовий запис пов'язаний з визначеною категорією. Архітектура програми повинна забезпечувати два основні сценарії роботи, навчання моделі на розміченому датасеті та практичне прогнозування категорії для нового тексту. Розділення зазначених сценаріїв дає змогу уникнути надмірного навантаження на користувацький інтерфейс, оскільки навчання моделі є тривалішим процесом, а класифікація одного тексту має виконуватися швидко та без повторного тренування нейромережі.

Програмна система розглядається як сукупність функціональних модулів, пов'язаних єдиним потоком даних. Перший модуль відповідає за підготовку датасеті, аналіз категорій, навчання нейромережевої моделі та збереження службових об'єктів. Другий модуль використовує вже навчену модель, токенизатор і таблицю відповідності класів для опрацювання нового тексту. За рахунок такого розподілу система може використовуватись як дослідницький інструмент для оцінювання якості моделі та як прикладний застосунок для автоматичного розподілу текстових записів за категоріями [30]. Функціональні модулі програмної системи класифікації текстів наведено в таблиці 2.10.

Таблиця 2.10 – Функціональні модулі програмної системи класифікації текстів

<i>Модуль системи</i>	<i>Призначення</i>	<i>Основний результат</i>
Модуль навчання	Підготовка розміченого корпусу, навчання моделі, розрахунок метрик	Навчена модель, токенизатор, таблиця класів, звіт якості
Модуль прогнозування	Класифікація нового текстового запису через збережену модель	Прогнозована категорія та рівень упевненості
Модуль збереження	Фіксація службових об'єктів, параметрів і результатів	Набір файлів для повторного запуску без нового навчання
Модуль оцінювання	Перевірка якості класифікації на тестовій вибірці	Accuracy, precision, recall, F1-score і матриця помилок

У режимі навчання, система отримує на вхід розмічений датасет текстів. Вхідний файл повинен містити щонайменше два поля, текстовий запис і категорію. Модуль перевірки структури визначає наявність порожніх значень, некоректних міток, дубльованих записів та надмірно коротких фрагментів. Після перевірки виконується аналіз розподілу категорій, оскільки нерівномірність класів може впливати на якість навчання. Результати аналізу використовуються для формування ваг класів або для фіксації проблеми дисбалансу в підсумковому звіті.

Кодування категорій виконується окремим модулем, який перетворює текстові назви класів на числові мітки. Одночасно формується таблиця зворотного відображення, потрібна для коректного виведення результату користувачеві. Якщо модель прогнозує числове значення, інтерфейс не повинен показувати код класу, оскільки користувач очікує зрозумілу назву категорії. Саме тому таблиця відповідності між кодами та назвами класів є обов'язковим службовим об'єктом системи [30].

Стратифікований поділ датасету є важливою частиною архітектури, оскільки він зберігає пропорції класів в навчальній, валідаційній та тестовій вибірках. Без стратифікації частина категорій може бути недостатньо представлена під час навчання або підсумкової перевірки. В результаті, модель

здатна демонструвати завищену загальну точність, але слабо працювати з окремими класами. Тому модуль поділу даних має фіксувати випадковий стан, щоб експеримент можна було повторити з однаковими умовами.

Після поділу датасету токенизатор навчається лише на навчальній вибірці. Це запобігає потраплянню інформації з тестових даних до словника ще до оцінювання. Далі токенизатор застосовується до навчальної, валідаційної та тестової частин за однаковими правилами. Отримані послідовності індексів вирівнюються до заданої довжини за допомогою padding і truncation. Компонентна архітектура програмної системи класифікації текстів наведена на рисунку 2.2.



Рисунок 2.2 – Компонентна архітектура програмної системи класифікації текстів

Компонентна діаграма показує, що навчання та прогнозування не змішуються в один нероздільний процес. Ліва частина відповідає за навчання,

де датасет проходить перевірку, кодування класів, поділ, токенизацію, навчання та оцінювання. Права частина відповідає за практичне використання, де новий текст передається до модуля прогнозування, а результат формується на основі збереженої моделі та службових об'єктів [30].

В режимі прогнозування користувач вводить новий текст через інтерфейс. Програма завантажує збережену модель, токенизатор, таблицю класів і параметр максимальної довжини послідовності. Далі текст проходить ті самі етапи підготовки, які використовувались під час навчання, а саме, нормалізацію, токенизацію, індексацію та вирівнювання довжини. Після цього нейромережева модель формує softmax-вектор, де кожне значення відповідає ймовірності належності тексту до окремої категорії.

Інтерфейс повинен відображати не лише прогнозовану категорію, але й рівень впевненості. Якщо найбільша ймовірність має низьке значення, система може показувати повідомлення про невпевнений прогноз. Для користувача це важливо, оскільки текст інколи може містити ознаки кількох тематичних груп або не відповідати жодній категорії з навчального датасету. Додаткове відображення ймовірностей за класами підвищує інформативність результату та допомагає зрозуміти поведінку моделі.

Файлова структура програмного продукту повинна підтримувати розподіл логіки навчання, прогнозування та інтерфейсу. Головний файл запуску може відповідати за відкриття вікна користувача, тоді як навчальний сценарій необхідно винести в окремий файл. Окремі модулі мають містити функції для підготовки даних, створення моделі, оцінювання та прогнозування.

Логіка роботи навчального сценарію починається з завантаження датасету з каталогу data/. Після перевірки структури виконується кодування міток класів та стратифікований поділ. Далі токенизатор формує словник за навчальною частиною корпусу, а тексти перетворюються на числові послідовності. Нейромережева модель навчається на навчальній вибірці з контролем якості за валідаційними даними. Після завершення навчання виконується оцінювання на тестовій частині, а модель і службові об'єкти

зберігаються в каталозі `models/`. Послідовність класифікації нового текстового запису наведена на рисунку 2.3.



Рисунок 2.3 – Послідовність класифікації нового текстового запису

Користувач не взаємодіє з внутрішніми параметрами нейромережі, а лише подає текст та отримує результат. Інтерфейс звертається до збережених об'єктів, оскільки для правильного прогнозування потрібні той самий токенизатор і та сама таблиця класів, які застосовувались під час навчання. Після формування softmax-вектора інтерпретатор визначає категорію з найбільшою ймовірністю та повертає її в зрозумілому вигляді.

Оцінювання системи виконується після навчання моделі. Програма застосовує модель до тестової вибірки, порівнює прогнозовані мітки з еталонними та формує звіт. До звіту мають входити accuracy, precision, recall, F1-score, macro average, weighted average і матриця помилок. Macro average потрібен для перевірки рівномірності якості за категоріями, а weighted average дозволяє врахувати кількість прикладів в кожному класі. Матриця помилок показує, між якими тематичними групами виникає найбільше неправильних прогнозів. Вихідні результати роботи програмної системи наведено в таблиці 2.11.

Таблиця 2.11 – Вихідні результати роботи програмної системи

<i>Результат</i>	<i>Формат збереження</i>	<i>Призначення</i>
Навчена модель	Файл моделі Keras	Повторне прогнозування без нового навчання
Токенізатор	Серіалізований файл або JSON	Однакове кодування навчальних і нових текстів
Таблиця класів	JSON або CSV	Перетворення числового прогнозу на назву категорії
Параметри підготовки	JSON-файл	Збереження максимальної довжини та розміру словника
Звіт метрик	TXT, CSV або JSON	Фіксація підсумкової якості моделі
Матриця помилок	PNG або CSV	Аналіз неправильних прогнозів
Графіки навчання	PNG	Відображення зміни accuracy і loss за епохами

Після запуску інтерфейсу користувач вводить текст в спеціальне поле, натискає кнопку класифікації та отримує назву категорії. За потреби вікно може додатково відображати рівень впевненості та кілька найімовірніших класів. Якщо модель або токенизатор не знайдені, інтерфейс повинен показати повідомлення про потребу попереднього навчання. Якщо поле введення порожнє, система має повідомити про неможливість класифікації без текстових даних.

Запропонована архітектура орієнтована саме на задачу автоматичного розподілу текстів за категоріями. Вона враховує наявність розміченого датасету, числове кодування класів, стратифікований поділ вибірок, збереження службових об'єктів та ймовірнісну інтерпретацію результату.

Висновки до розділу 2

В другому розділі сформовано проєктну основу програмної системи для автоматичної класифікації текстових записів з використанням нейронної мережі. Задачу подано як процес контрольованого навчання, де кожному текстовому запису відповідає наперед визначена категорія, а модель повинна навчитися встановлювати зв'язок між мовними ознаками та класами. Визначено вхідні дані, вихідний результат, структуру вибірок і загальну логіку перетворення тексту на числове представлення.

Обґрунтовано методи підготовки датасету до класифікації, зокрема перевірку структури даних, кодування категорій, стратифікований поділ, токенизацію, обробку невідомих слів, padding, truncation і збереження службових об'єктів.

Розроблено архітектуру моделі багатокласової класифікації, яка містить вхідний шар, embedding-шар, блок узагальнення ознак, повнозв'язний класифікаційний шар, dropout і softmax-вихід. Визначено метрики оцінювання

ефективності, серед яких accuracy, precision, recall, F1-score, validation loss і матриця помилок. Також спроектовано архітектуру програмної системи з окремими режимами навчання, прогнозування, аналізу тексту, пакетної обробки та формування звітів.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМИ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ

3.1 Формування набору текстових даних та виконання попередньої обробки текстової інформації

Практична реалізація моделі класифікації текстів потребує підготовки розміченого датасету, де кожен текстовий запис має відповідну категорію. Для навчання нейромережевої моделі в роботі обрано датасет новинних повідомлень, оскільки новинні тексти мають чітку тематичну належність, відносно усталену структуру та достатню кількість лексичних ознак для розпізнавання категорій. Вибір новинного датасету також пояснюється тим, що вказаний тип даних добре підходить для демонстрації задачі багатокласової класифікації, де один текст має бути віднесений лише до однієї тематичної групи.

Для програмної реалізації використано набір текстових даних з п'ятьма тематичними категоріями, а саме, бізнес, розваги, політика, спорт і технології. Кожен запис датасету складається з текстового фрагменту та мітки класу. Текстовий фрагмент є вхідним об'єктом для моделі, а мітка класу виконує роль правильної відповіді під час контрольованого навчання. Наявність розмітки дозволяє моделі встановлювати залежність між мовними ознаками тексту та відповідною тематичною категорією. Структура набору текстових даних для класифікації наведена в таблиці 3.1.

Поле `text` використовується для попередньої обробки і токенизації, а поле `category` потрібне для формування міток класів. Поле `label` створюється під час підготовки даних автоматично, оскільки нейромережева модель не працює безпосередньо з текстовими назвами категорій. Для відображення результату користувачеві зберігається зворотна відповідність між числовими кодами та назвами класів.

Таблиця 3.1 – Структура набору текстових даних для класифікації

<i>Поле набору даних</i>	<i>Зміст поля</i>	<i>Призначення в програмній системі</i>
id	Унікальний номер текстового запису	Забезпечення зручного обліку прикладів
text	Текст новинного повідомлення	Основний вхідний матеріал для класифікації
category	Назва тематичної категорії	Еталонна мітка для навчання моделі
label	Числовий код категорії	Формат мітки, придатний для функції втрат

Перед навчанням моделі виконано аналіз кількості прикладів в кожній категорії. Аналіз розподілу класів є важливим, оскільки надмірна перевага однієї категорії може вплинути на поведінку класифікатора. Якщо модель бачить значно більше прикладів одного класу, вона може частіше прогнозувати саме його, навіть коли текст належить до іншої тематики. Для зменшення ризику викривленого навчання під час поділу датасету використовується стратифікація, яка зберігає пропорції категорій в навчальній, валідаційній та тестовій частинах. Розподіл текстових записів за тематичними категоріями наведено в таблиці 3.2.

Таблиця 3.2 – Розподіл текстових записів за тематичними категоріями

<i>Категорія</i>	<i>Кількість текстових записів</i>	<i>Частка в датасеті, %</i>
Бізнес	510	22,9
Розваги	386	17,3
Політика	417	18,7
Спорт	511	23,0
Технології	401	18,1
Загалом	2225	100,0

Найбільшу кількість прикладів мають категорії «спорт» і «бізнес», а найменшу – категорія «розваги». Різниця між кількістю записів в категоріях не

є критичною, однак під час оцінювання моделі потрібно аналізувати не лише загальну точність, але й precision, recall та F1-score для кожного класу окремо.

Попередня обробка текстової інформації виконується для приведення датасету до єдиного формату. Початкові записи можуть містити зайві пробіли, різні варіанти регістру, розділові знаки, службові символи або фрагменти, які не мають суттєвого значення для визначення тематики. Без підготовки модель отримувала б нерівномірні та шумові ознаки, що знижувало б якість навчання. Обробка має бути помірною, оскільки надмірне вилучення елементів може призвести до втрати корисної інформації, особливо в новинних текстах, де власні назви, числа та терміни можуть бути важливими для розпізнавання категорії. Етапи попередньої обробки текстової інформації наведено в таблиці 3.3.

Таблиця 3.3 – Етапи попередньої обробки текстової інформації

<i>Етап обробки</i>	<i>Зміст виконаної дії</i>	<i>Результат для моделі</i>
Перевірка порожніх записів	Виявлення рядків без тексту або категорії	Усунення некоректних прикладів
Видалення дублікатів	Пошук повторюваних текстових фрагментів	Зменшення ризику штучного завищення якості
Нормалізація регістру	Приведення літер до нижнього регістру	Скорочення кількості дублювань у словнику
Очищення службових символів	Вилучення технічних позначок і зайвих пробілів	Зменшення шуму в корпусі
Кодування категорій	Перетворення назв класів на числові коди	Підготовка міток для навчання
Стратифікований поділ	Формування train, validation і test-вибірок	Збереження пропорцій категорій
Токенізація	Перетворення текстів на послідовності токенів	Підготовка до числового представлення
Вирівнювання довжини	Padding і truncation послідовностей	Формування єдиного формату входу

Перевірка порожніх записів виконується на початку, оскільки відсутність тексту або категорії унеможливило б коректне навчання. Записи без текстової частини не містять ознак для класифікації, а записи без мітки не можуть використовуватися в контрольованому навчанні. Дублікати також потребують видалення, оскільки повторення однакових текстів може призвести до ситуації, коли модель фактично запам'ятовує окремі приклади, а не вивчає загальні закономірності категорій.

Нормалізація реєстру дозволяє зменшити кількість варіантів написання однакових слів. Наприклад, слова, написані з великої літери на початку речення, після нормалізації не створюють окремих одиниць словника. Очищення службових символів спрямоване на усунення випадкових елементів, які не допомагають розпізнавати тему.

Мітки категорій кодуються в числовий формат за допомогою таблиці відповідності. Наприклад, категорія «бізнес» може отримати код 0, «розваги» – код 1, «політика» – код 2, «спорт» – код 3, «технології» – код 4. Після кодування формується службовий об'єкт, який зберігає відповідність між кодом і назвою класу. Під час прогнозування модель повертає числовий результат, а інтерфейс програми перетворює його на зрозумілу для користувача назву категорії.

Після підготовки, датасет поділяється на навчальну, валідаційну та тестову вибірки. Навчальна частина використовується для налаштування ваг нейромережі. Валідаційна частина потрібна для контролю поведінки моделі під час епох навчання. Тестова частина застосовується для фінальної перевірки якості після завершення навчання. Стратифікований поділ дозволяє зберегти співвідношення категорій в кожній частині датасету, що робить оцінювання більш надійним.

Токенізація виконується після поділу датасету, причому словник формується лише за навчальною вибіркою. Наведений порядок потрібен для запобігання витоку інформації з тестової частини. Якщо словник формувати на всьому датасеті, модель опосередковано отримає відомості про тестові записи

ще до оцінювання. Після навчання токенизатора тексти всіх вибірок перетворюються на послідовності числових індексів за однаковими правилами.

Довжина текстів в датасеті є нерівномірною, тому перед поданням до нейромережі послідовності вирівнюються. Короткі записи доповнюються службовими нульовими значеннями, а надмірно довгі обрізаються до встановленої межі. Максимальна довжина добирається на основі аналізу розподілу кількості токенів в текстах. Завдяки вирівнюванню всі приклади мають однакову форму, що є обов'язковою умовою для пакетного навчання нейромережевої моделі. Параметри підготовки текстів до нейромережевого навчання наведено в таблиці 3.4

Таблиця 3.4 – Параметри підготовки текстів до нейромережевого навчання

<i>Параметр</i>	<i>Обране значення</i>	<i>Обґрунтування</i>
Максимальна кількість слів у словнику	10000	Дозволяє зберегти найуживаніші токени й обмежити розмір embedding-шару
Максимальна довжина послідовності	200 токенів	Покриває основну частину новинних текстів без надмірного збільшення входу
Службовий токен невідомого слова	OOV	Забезпечує обробку слів, відсутніх у навчальному словнику
Тип вирівнювання	Padding після тексту	Зберігає початкову частину повідомлення без зміщення змісту
Тип обрізання	Truncation після встановленої межі	Запобігає перевищенню допустимої довжини входу

Після завершення підготовки кожен текстовий запис набуває вигляду числової послідовності фіксованої довжини, а кожна категорія має числову мітку. В результаті формується набір масивів, придатних для подання до нейромережевої моделі. Окремо зберігаються токенизатор, таблиця класів і параметр максимальної довжини послідовності. Вказані службові об'єкти потрібні не лише під час навчання, але й під час класифікації нових текстів в готовій програмі.

3.2 Реалізація моделі для автоматичного розподілу текстів за категоріями

Реалізація програмної моделі для автоматичного розподілу текстів за категоріями виконана у форматі локального вебзастосунку TextCategory AI, який відкривається в браузері та працює на основі серверної частини Python. Вибір браузерного формату пояснюється зручністю демонстрації результатів класифікації, можливістю виконання скриншотів для практичного розділу, а також, простим запуском програми на персональному комп'ютері без потреби розміщення системи на зовнішньому сервері. Серверна частина відповідає за підготовку тексту, завантаження збереженої моделі, виконання прогнозу, повторне навчання та формування звітних матеріалів, а користувацький інтерфейс забезпечує введення або завантаження текстів і перегляд результатів.

В системі реалізовано роботу з розміченим набором текстових даних, кодування категорій, стратифікований поділ корпусу, токенизацію, обробку невідомих слів через службовий OOV-токен, вирівнювання довжини послідовностей, нейромережеву модель з embedding-шаром, softmax-вихід, а також, базову модель TF-IDF і Logistic Regression для порівняння. За рахунок такого набору компонентів, програма виконує не лише класифікацію окремого тексту, але й демонструє повний цикл реалізації моделі машинного навчання.

Основним етапом реалізації є підготовка текстів до нейромережевої моделі. Спочатку програма завантажує розмічений датасет з каталогу data/, перевіряє наявність текстів і категорій, вилучає порожні записи, а потім виконує кодування міток класів. Назви категорій перетворюються на числові коди, які використовуються під час навчання, а таблиця відповідності між кодами та назвами зберігається для подальшого виведення результату в інтерфейсі. Завдяки цьому користувач бачить не числовий код, а зрозумілу назву тематичної категорії.

Після перевірки датасету виконується стратифікований поділ на навчальну, валідаційну та тестову частини. Стратифікація потрібна для збереження співвідношення категорій в кожній вибірці, оскільки випадковий поділ без контролю класів може спотворити результати навчання. Токенізатор навчається лише на навчальній частині датасету, щоб інформація з тестової вибірки не впливала на формування словника. Для невідомих слів використовується службовий OOV-токен, а всі числові послідовності вирівнюються до встановленої довжини за допомогою padding і truncation. Реалізовані етапи підготовки текстів до класифікації наведено в таблиці 3.5.

Таблиця 3.5 – Реалізовані етапи підготовки текстів до класифікації

<i>Етап реалізації</i>	<i>Програмне виконання</i>	<i>Результат</i>
Завантаження датасету	Зчитування CSV-файлу з текстами та категоріями	Табличний набір прикладів
Перевірка структури	Контроль наявності текстового поля та мітки класу	Відібраний набір коректних записів
Кодування класів	Перетворення назв категорій на числові коди	Масив міток для навчання
Стратифікований поділ	Розподіл датасету на train, validation і test	Вибірки зі збереженням пропорцій класів
Навчання токенизатора	Формування словника за навчальною вибіркою	Об'єкт токенизатора для повторного використання
Обробка OOV-токенів	Присвоєння службового індексу невідомим словам	Стійкість до нових слів під час прогнозування
Padding і truncation	Доповнення або обрізання послідовностей	Однакова довжина входу для нейромережі

Нейромережева модель реалізована як послідовність шарів, орієнтованих на багатокласову класифікацію. На вхід надходить числова послідовність токенів фіксованої довжини. Embedding-шар перетворює індекси токенів на щільні векторні представлення, які краще підходять для виявлення тематичних закономірностей. Після цього використовується шар узагальнення ознак, який

формує компактне представлення всього текстового запису. Класифікаційний блок містить повнозв'язний шар і dropout-рівень, а вихідний шар з функцією softmax повертає ймовірності належності тексту до кожної категорії.

Сторінка навчання нейромережевої моделі у вебзастосунку наведена на рисунку 3.1.

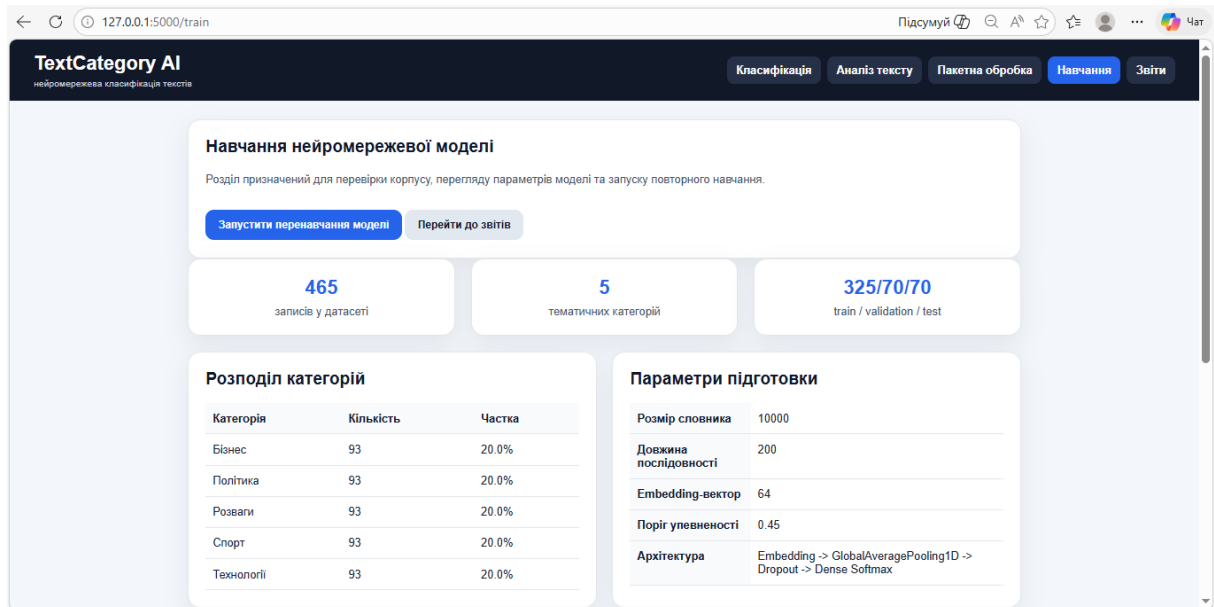


Рисунок 3.1 – Сторінка навчання нейромережевої моделі у вебзастосунку

Навчання моделі запускається через відповідний маршрут вебзастосунку. Після натискання кнопки перенавчання програма виконує повний цикл підготовки даних, формує словник, створює нейромережеву архітектуру, запускає навчання та зберігає результати. Для зниження ризику перенавчання використовується dropout, контроль валідаційної функції втрат і механізм раннього зупинення. Якщо якість на валідаційній вибірці перестає покращуватись, навчання завершується, а в системі зберігається стан моделі з найкращими показниками.

Крім основної нейромережевої моделі, в програмі реалізовано базовий класифікатор на основі TF-IDF і Logistic Regression. Його призначення полягає в отриманні контрольного результату для порівняння з embedding-моделлю. Під

час навчання обидві моделі оцінюються на тестовій вибірці, а отримані результати зберігаються в каталозі results/. Об'єкти, які зберігаються після навчання моделі наведено в таблиці 3.6.

Таблиця 3.6 – Об'єкти, які зберігаються після навчання моделі

<i>Збережений об'єкт</i>	<i>Призначення</i>
Файл нейромережевої моделі	Повторне використання класифікатора без нового навчання
Токенізатор	Однакове кодування навчальних і нових текстів
Кодувальник міток	Перетворення між кодами класів і назвами категорій
Таблиця класів	Відображення прогнозованого класу у зрозумілому вигляді
Параметри моделі	Збереження максимальної довжини й розміру словника
Звіт метрик	Аналіз якості класифікації на тестовій вибірці
Матриця помилок	Виявлення класів, між якими виникають неправильні прогнози
Графіки навчання	Візуальна оцінка зміни loss і accuracy за епохами

Після завершення навчання модель використовується в режимі прогнозування. Користувач може ввести текст вручну або завантажити текстовий файл через інтерфейс. Програма застосовує до нового тексту ті самі правила підготовки, які використовувалися під час навчання, нормалізацію, токенизацію, заміну невідомих слів службовим індексом, вирівнювання довжини та передавання числової послідовності до моделі. Вихідний softmax-вектор інтерпретується як розподіл імовірностей за класами, а категорія з найбільшим значенням виводиться як прогноз.

В режимі прогнозування користувач отримує назву категорії, рівень впевненості та розподіл імовірностей за доступними класами. Додатково в застосунку передбачено сторінку «Аналіз тексту», де відображаються етапи підготовки введеного запису, а саме, очищений текст, токени, кількість невідомих одиниць і фрагмент числової послідовності. Сторінка аналізу текстового запису перед класифікацією наведена на рисунку 3.2.

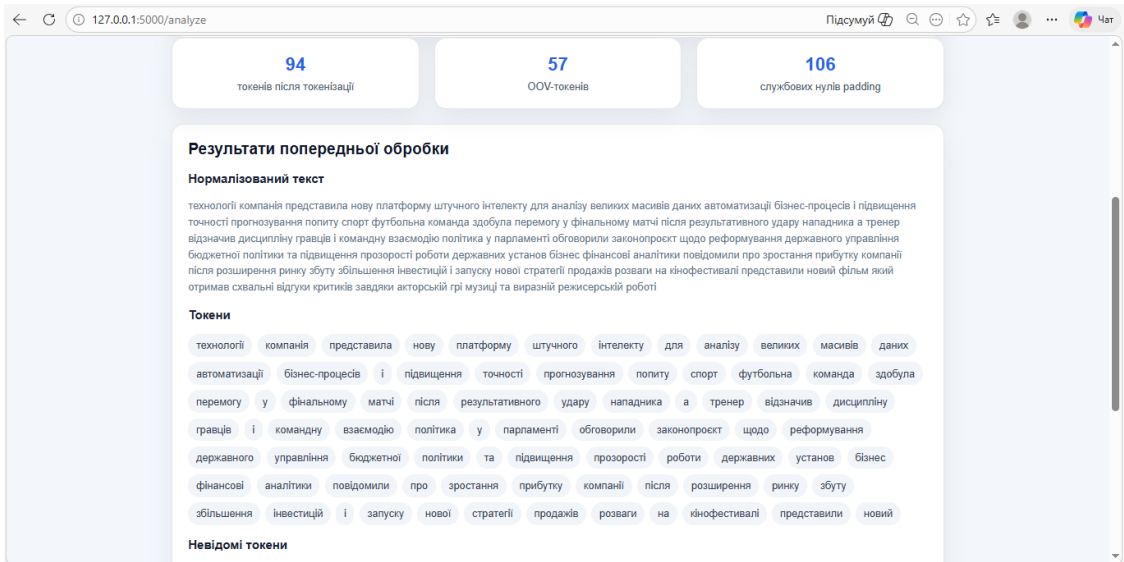


Рисунок 3.2 – Сторінка аналізу текстового запису перед класифікацією

Окремою функцією реалізовано пакетну обробку текстів. Користувач може вставити кілька текстових записів або завантажити файл з набором фрагментів, після чого система виконує класифікацію кожного запису окремо. Результати подаються у вигляді таблиці з прогнозованими категоріями та рівнями впевненості. Сторінка пакетної класифікації текстових записів наведена на рисунку 3.3.

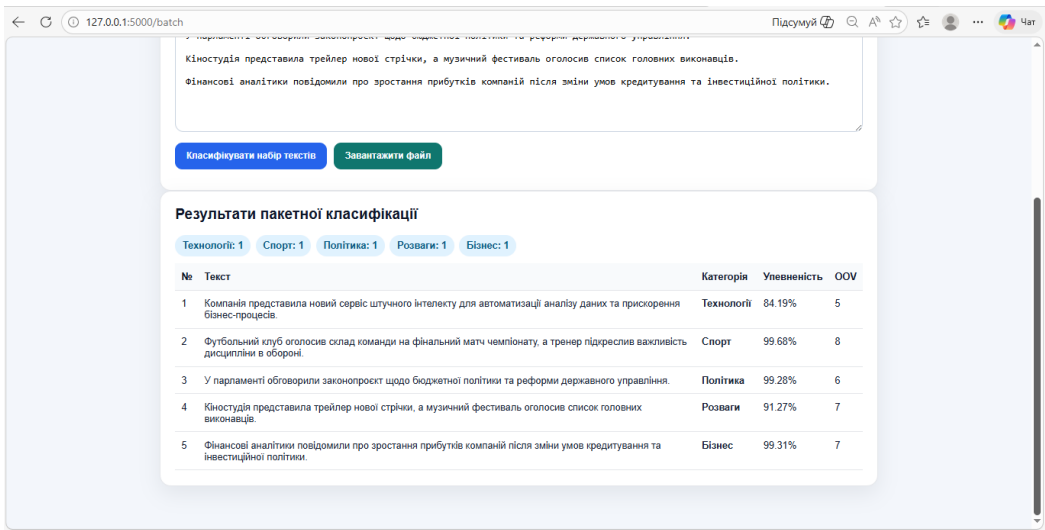


Рисунок 3.3 – Сторінка пакетної класифікації текстових записів

Реалізована модель також формує матеріали для подальшого аналізу ефективності. На сторінці «Звіти» можна переглянути порівняння нейромережевої моделі з базовою Logistic Regression, значення метрик за категоріями, матрицю помилок і графіки навчання. Сторінка звітів з результатами оцінювання моделі наведена на рисунку 3.4.

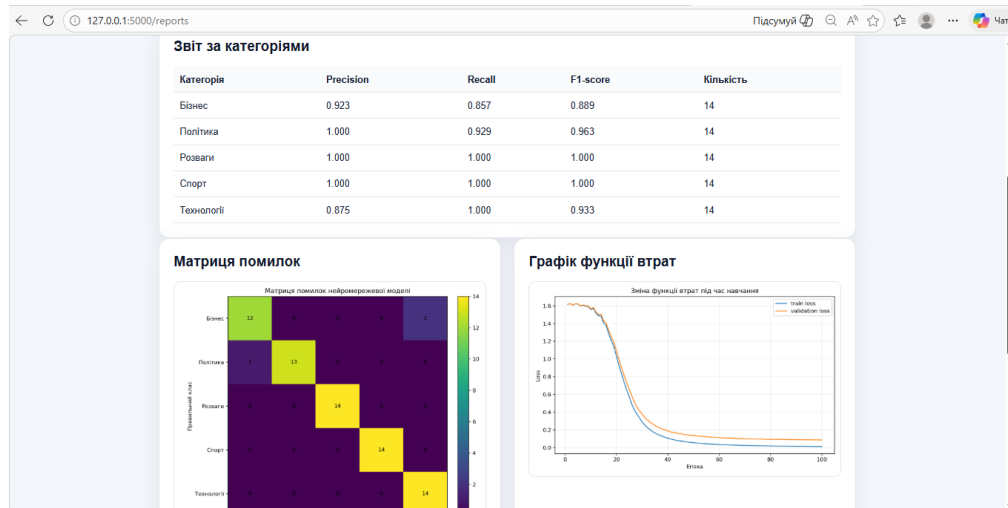


Рисунок 3.4 – Сторінка звітів з результатами оцінювання моделі

Реалізований програмний продукт відповідає поставленій задачі автоматичного розподілу текстів за тематичними категоріями та створює основу для подальшого аналізу результатів класифікації, опису інтерфейсу та тестування системи.

3.3 Застосування нейромережі для класифікації на практиці та аналіз отриманих результатів

Практичне застосування нейромережевої моделі реалізовано через вебінтерфейс програмного продукту TextCategory AI. Після запуску локального

сервера користувач відкриває застосунок в браузері, вводить текстовий запис або завантажує файл з текстом, після чого система виконує класифікацію та відображає прогнозовану категорію. Взаємодія з моделлю не потребує від користувача знання внутрішньої структури нейронної мережі, оскільки всі етапи попередньої обробки, токенизації, перетворення тексту на числову послідовність і подання до моделі виконуються автоматично.

Основний сценарій практичного використання полягає у введенні одного текстового фрагмента на сторінці класифікації. Після натискання кнопки класифікації серверна частина застосунку передає текст до модуля підготовки, застосовує збережений токенизатор, вирівнює довжину послідовності та подає отриманий масив до нейромережевої моделі. Сторінка класифікації текстового запису у вебзастосунку TextCategory AI наведена на рисунку 3.5.

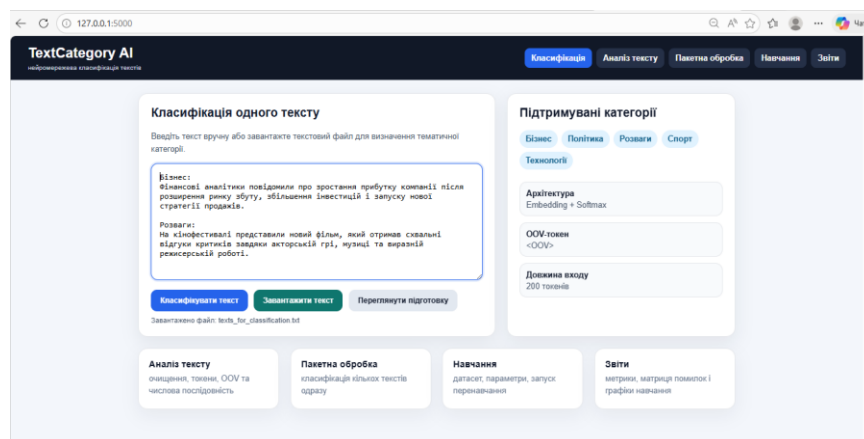


Рисунок 3.5 – Сторінка класифікації текстового запису у вебзастосунку TextCategory AI

Після запуску класифікації система повертає назву прогнозованої категорії та рівень упевненості. Рівень впевненості відображає найбільше значення у векторі ймовірностей, сформованому вихідним softmax-шаром. Наприклад, якщо для категорії «Технології» отримано найвище значення, текст відноситься до технологічної тематики, а користувач додатково бачить числовий показник впевненості. Наявність розподілу ймовірностей за всіма

категоріями дозволяє оцінити, наскільки однозначним є прогноз. Результат класифікації текстового запису з відображенням категорії та рівня впевненості наведено на рисунку 3.6.

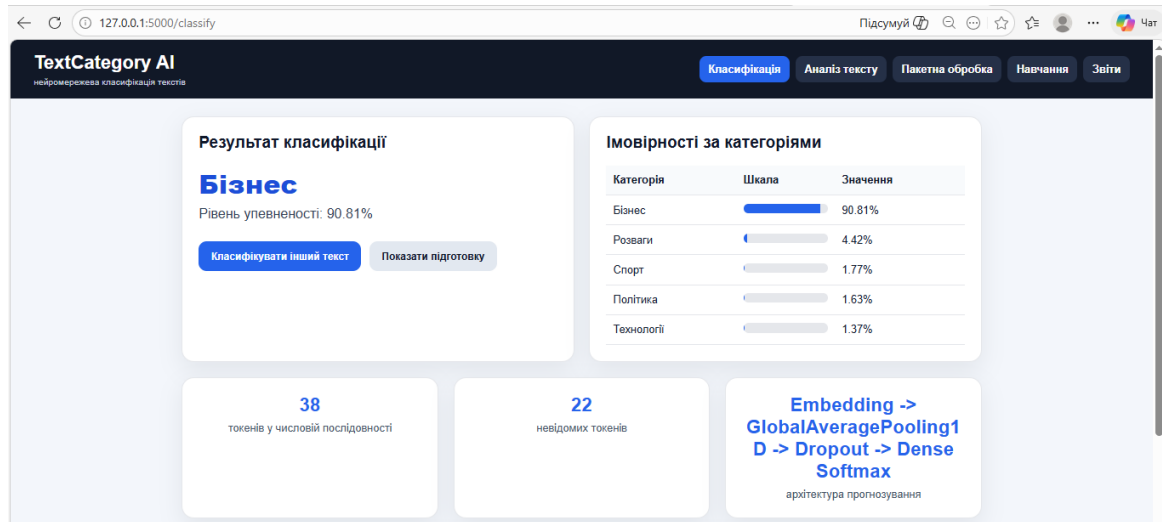


Рисунок 3.6 – Результат класифікації текстового запису з відображенням категорії та рівня впевненості

Практична перевірка роботи моделі виконувалась на текстах, які належать до п'яти тематичних груп: бізнес, політика, розваги, спорт і технології. Для кожної категорії обирались короткі фрагменти, що містять характерну лексику відповідної тематики. Під час класифікації система аналізувала не окреме ключове слово, а весь текстовий запис, перетворений в числову послідовність. Завдяки цьому модель могла враховувати сукупність лексичних ознак, а не лише наявність одного терміну.

Проміжний аналіз тексту підтверджує, що введений запис проходить всі етапи, описані у проєктній частині роботи. Спочатку виконується нормалізація, після чого текст розбивається на токени. Токени, наявні у словнику, перетворюються на відповідні індекси, а невідомі слова отримують службове позначення OOV. Далі послідовність приводиться до встановленої довжини, що забезпечує сумісність з вхідним шаром нейромережі.

Пакетна класифікація показує, що програмний продукт може застосовувати навчену модель не лише для окремого запиту, а й для групи текстів. Всі записи проходять однаковий шлях підготовки, після чого для кожного формується власний softmax-вектор. Якщо текст має чітку тематичну спрямованість, одна категорія отримує суттєво вищу ймовірність. Якщо текст містить лексику з кількох тематичних груп, різниця між найвищими ймовірностями може бути меншою, що свідчить про неоднозначність класифікації. Інтерпретація рівня впевненості під час класифікації наведена в таблиці 3.7.

Таблиця 3.7 – Інтерпретація рівня впевненості під час класифікації

<i>Значення найбільшої ймовірності</i>	<i>Характеристика прогнозу</i>	<i>Інтерпретація результату</i>
Понад 0,80	Висока впевненість	Текст має виразні ознаки прогнозованої категорії
Від 0,60 до 0,80	Помірна впевненість	Текст переважно належить до обраної категорії, але має окремі ознаки інших класів
Від 0,40 до 0,60	Низька впевненість	Тематична належність не є однозначною
Менше 0,40	Невпевнений прогноз	Текст може не відповідати жодній категорії або містити недостатньо ознак

Аналіз рівня впевненості потрібен для коректного трактування результатів класифікації. Високе значення найбільшої ймовірності не гарантує абсолютну правильність прогнозу, але показує, що модель виявила достатньо сильну відповідність між текстом та певним класом. Низьке значення свідчить про потребу додаткової перевірки, особливо якщо текст короткий, містить загальні слова або поєднує кілька тематик.

Для оцінювання практичних результатів у програмному продукті передбачено сторінку звітів. На ній відображається порівняння нейромережевої моделі з базовою Logistic Regression, метрики за категоріями, матриця помилок і графіки навчання. Порівняння з базовою моделлю дає змогу перевірити, чи

забезпечує embedding-підхід прийнятну якість порівняно з класичним TF-IDF-представленням. Матриця помилок показує, між якими категоріями виникають неправильні прогнози, а графіки навчання дозволяють оцінити стабільність процесу оптимізації.

Практичний аналіз результатів виконується за кількома напрямками. Загальна точність дає уявлення про частку правильних прогнозів, але не розкриває поведінку моделі за окремими категоріями. Precision показує, наскільки надійним є віднесення текстів до певного класу. Recall відображає здатність моделі знаходити всі тексти відповідної категорії. F1-score узгоджує precision і recall, тому є важливим показником для порівняння якості за різними класами. Якщо одна категорія має нижчий F1-score, потрібно переглянути приклади цієї групи або збільшити кількість навчальних записів відповідної тематики.

Матриця помилок дозволяє визначити характер неправильних прогнозів. Діагональні елементи відповідають правильно класифікованим текстам, а значення поза діагоналлю відображають випадки плутанини між категоріями. Якщо модель часто плутає політичні та бізнесові тексти, це може пояснюватися спільною лексикою, пов'язаною з державним регулюванням, економічними рішеннями або фінансовою політикою. Якщо помилки виникають між розвагами й технологіями, причиною може бути наявність текстів про цифрові медіа, платформи або онлайн-сервіси.

Графіки навчання показують, як змінювались показники моделі під час епох оптимізації. Якщо значення функції втрат на навчальній та валідаційній вибірках зменшуються без різкого розходження, навчання можна вважати стабільним. Якщо навчальна точність зростає, а валідаційна якість погіршується, виникає ознака перенавчання. В реалізованій моделі для зменшення такого ризику застосовуються dropout і рання зупинка, яка дозволяє зберігати модель з кращою здатністю до узагальнення.

Нейромережа забезпечує автоматичний розподіл текстових записів за тематичними категоріями, відображає рівень впевненості та дає змогу

аналізувати проміжні етапи підготовки тексту. Додаткові сторінки пакетної обробки та звітів розширюють практичне значення програмного продукту, оскільки система може працювати як з одиничним текстом, так і з групою записів, а також формувати матеріали для оцінювання якості моделі. Отримані результати підтверджують працездатність обраної архітектури та створюють основу для подальшого опису інтерфейсу користувача й тестування системи.

3.4 Опис інтерфейсу користувача та детальна інструкція щодо використання програми

Інтерфейс програмного продукту TextCategory AI реалізовано у вигляді локального вебзастосунку, який відкривається через браузер після запуску серверної частини на комп'ютері користувача. Головною метою інтерфейсу є забезпечення зручної взаємодії з нейромережевою моделлю без потреби прямого звернення до програмного коду. Користувач працює з вебсторінками, кнопками, текстовими полями та файлами, а всі операції з підготовки тексту, завантаження моделі, токенизації, побудови числової послідовності та отримання прогнозу виконуються автоматично.

Запуск програми виконується з локальної папки програмного продукту. Для роботи застосунку передбачено файл start.bat, який активує потрібне середовище, встановлює або перевіряє залежності та запускає Flask-сервер.

Після відкриття вебзастосунку користувач бачить головну сторінку класифікації. Верхня навігаційна панель містить назву TextCategory AI та переходить до основних режимів роботи. Основний блок сторінки призначений для введення текстового запису, який необхідно класифікувати. Праворуч від робочої області відображаються підтримувані категорії, що допомагає користувачеві розуміти, до яких тематичних груп модель може віднести текст.

Інтерпретація рівня впевненості під час класифікації представлено на рисунку 3.7.

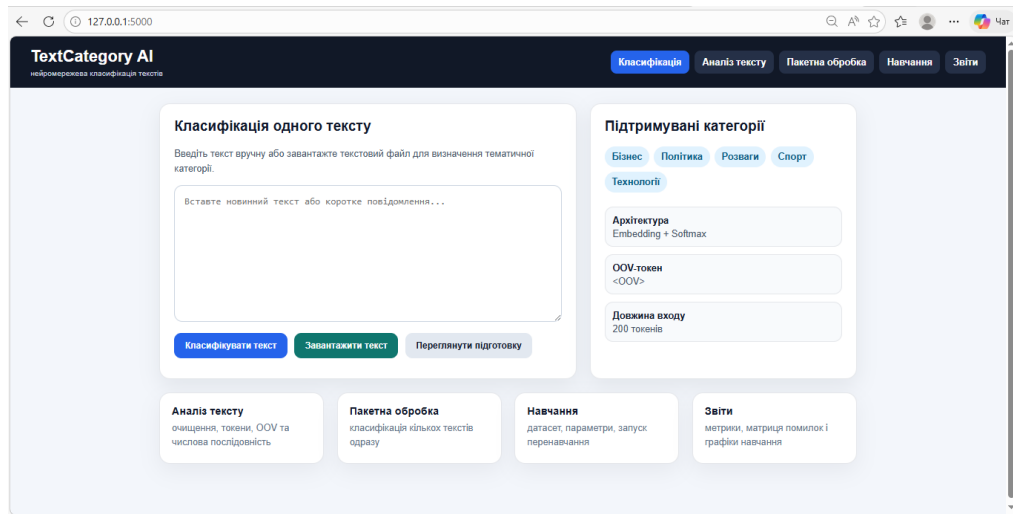


Рисунок 3.7 – Початкове вікно вебзастосунку TextCategory AI після запуску

Основний режим роботи передбачає ручне введення тексту. Користувач вставляє новинний фрагмент або коротке повідомлення в поле введення, після чого натискає кнопку класифікації. Програма передає текст до серверного модуля, виконує підготовку запису та повертає результат. Якщо потрібно класифікувати текст з зовнішнього файлу, передбачено кнопку завантаження. Після вибору текстового документу його вміст підставляється до поля введення, що зменшує кількість ручних дій і спрощує перевірку готових прикладів. Завантаження текстового файлу для подальшої класифікації наведено на рисунку 3.8.

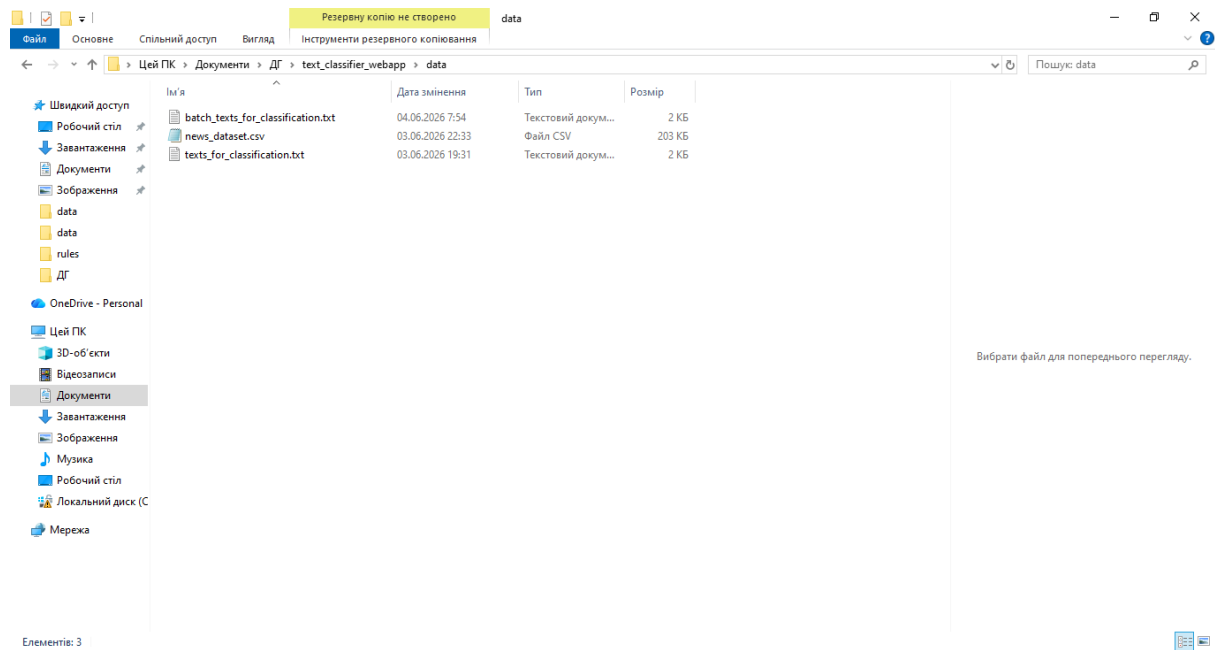


Рисунок 3.8 – Завантаження текстового файла для подальшої класифікації

Інтерфейс містить перевірку коректності введення. Якщо користувач намагається виконати класифікацію без тексту, система не запускає модель, а повідомляє про потребу заповнення поля. Даний механізм запобігає передаванню порожнього рядка до модуля прогнозування та підвищує стійкість програми до помилок користувача. Повідомлення виводиться без перезавантаження складних компонентів моделі, тому користувач може одразу виправити введення. Повідомлення системи про потребу введення тексту наведено на рисунку 3.9.

Основні функції інтерфейсу згруповано за практичними сценаріями. Сторінка класифікації використовується для роботи з одним текстом. Сторінка аналізу тексту показує службові етапи перетворення запису перед поданням до нейромережі. Пакетна обробка застосовується для класифікації групи текстів. Сторінка навчання використовується для повторного формування моделі на наявному датасеті, а сторінка звітів дає змогу переглянути результати оцінювання.

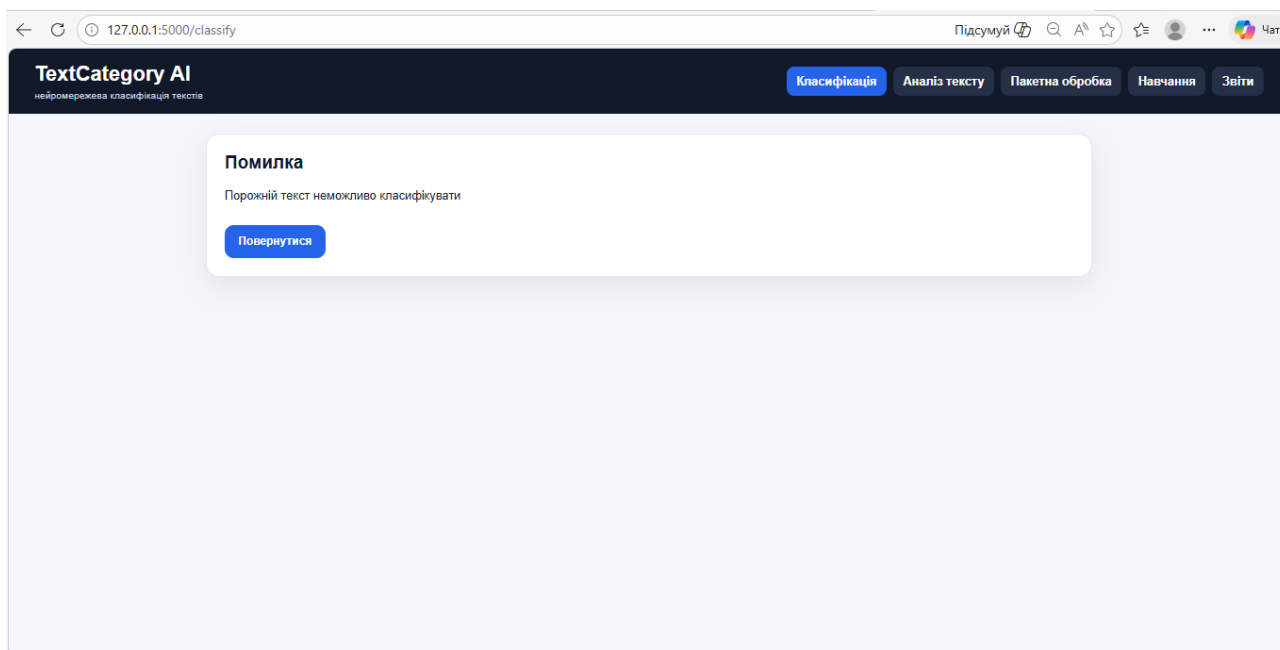


Рисунок 3.9 – Повідомлення системи про потребу введення тексту

Повторне навчання моделі виконується через кнопку «Перенавчити». Після її натискання серверна частина знову завантажує датасет, виконує перевірку структури, формує навчальну, валідаційну та тестову вибірки, навчає токенизатор, створює нейромережеву модель і зберігає оновлені об'єкти. Користувачеві не потрібно вручну запускати окремі Python-файли, оскільки сценарій навчання інтегровано в інтерфейс. Після завершення процесу застосунок показує повідомлення про успішне перенавчання. Повідомлення про успішне перенавчання нейромережевої моделі наведено на рисунку 3.10.

Під час використання програми важливо дотримуватись логіки підготовки вхідних матеріалів. Найкраще модель працює з текстами, які мають зрозумілу тематичну спрямованість і достатній обсяг змістовних слів. Надто короткі повідомлення можуть давати менш упевнені прогнози, оскільки в них недостатньо мовних ознак для впевненого розпізнавання категорії. Тексти з ознаками кількох тематик також можуть формувати близькі значення ймовірностей для різних класів.

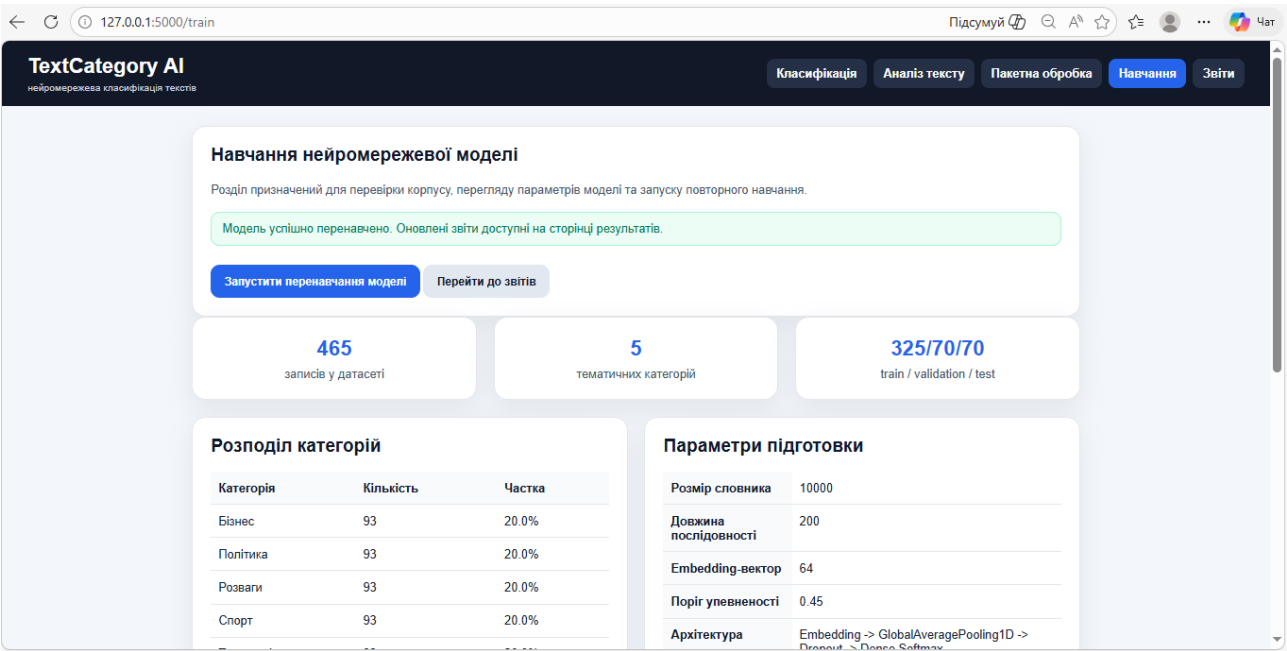


Рисунок 3.10 – Повідомлення про успішне перенавчання нейромережевої моделі

Для зручності перевірки в каталозі data передбачено файли з готовими текстами. Файл `texts_for_classification.txt` використовується для перевірки одиничного режиму, а файл `batch_texts_for_classification.txt` призначений для пакетної класифікації. Користувач може створювати власні ТХТ-файли, але бажано зберігати їх в кодуванні UTF-8, щоб уникнути некоректного відображення символів під час завантаження. Файл із текстами для перевірки роботи класифікатора наведено на рисунку 3.11.

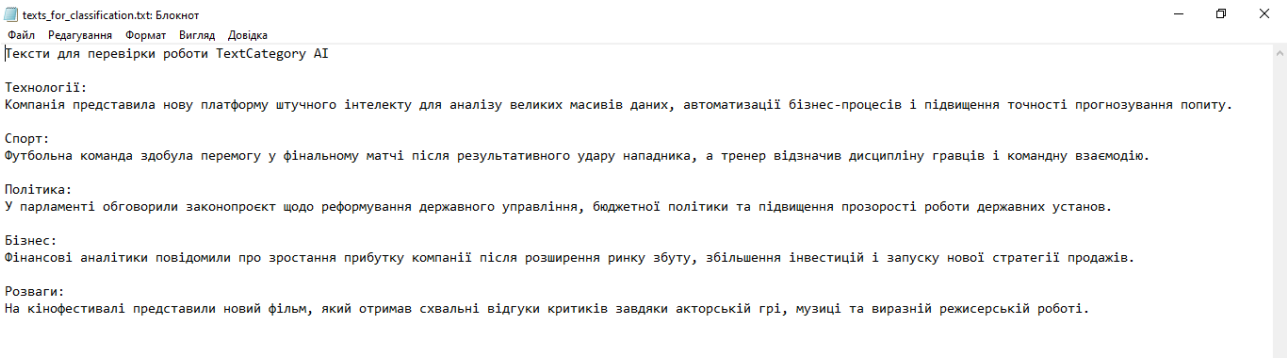


Рисунок 3.11 – Файл з текстами для перевірки роботи класифікатора

Інтерфейс вебзастосунку побудовано за принципом відокремлення користувацьких дій від внутрішньої логіки моделі. Користувач взаємодіє лише з полями, кнопками та сторінками, а серверна частина виконує всі обчислювальні операції.

3.5 Тестування системи та оцінка якості її функціонування

Тестування програмної системи TextCategory AI виконано для перевірки коректності роботи основних функцій, стійкості до помилкових дій користувача, правильності класифікації текстових записів і наявності службових результатів, потрібних для аналізу якості моделі. Перевірка охоплювала запуск вебзастосунку, введення тексту вручну, завантаження текстових файлів, класифікацію неоднозначних повідомлень, пакетну обробку, повторне навчання моделі, формування звітів і перевірку збережених файлів.

Під час тестування враховувалось, що програмний продукт має працювати в двох основних режимах. Перший режим пов'язаний з практичною класифікацією текстових записів, де користувач очікує швидке отримання категорії та рівня впевненості. Другий режим стосується дослідницької перевірки якості, де важливими є метрики, матриця помилок, графіки навчання та звітні файли. В обох режимах система повинна зберігати стабільність роботи, не переривати виконання через некоректне введення та повертати зрозумілий результат. Основні сценарії тестування програмної системи наведено в таблиці 3.8.

Таблиця 3.8 – Основні сценарії тестування програмної системи

<i>Тестовий сценарій</i>	<i>Перевірювана функція</i>	<i>Очікуваний результат</i>
Запуск локального сервера	Робота Flask-застосунку	Відкриття вебінтерфейсу за локальною адресою
Введення одного тексту	Класифікація окремого запису	Виведення категорії та рівня упевненості
Завантаження TXT-файла	Читання зовнішнього текстового документа	Підстановка тексту у поле введення
Порожнє поле введення	Контроль помилкових дій користувача	Повідомлення про потребу введення тексту
Неоднозначний текст	Перевірка ймовірнісного прогнозу	Виведення категорії з нижчим рівнем упевненості
Пакетна обробка	Класифікація групи записів	Таблиця результатів для кількох текстів
Повторне навчання	Оновлення моделі та службових об'єктів	Повідомлення про завершення навчання
Перегляд звітів	Оцінювання якості класифікації	Відображення метрик, графіків і матриці помилок

Функціональне тестування показало, що інтерфейс коректно реагує на стандартні дії користувача. При введенні змістового тексту система запускає класифікацію та повертає тематичну категорію. При завантаженні файлу текст автоматично потрапляє до робочого поля. Якщо користувач залишає поле порожнім, система не передає порожній рядок до моделі, а виводить повідомлення про потребу введення тексту. Наявність такої перевірки зменшує ймовірність помилки під час практичного використання.

Окрема увага приділялась неоднозначним текстам, які містять ознаки кількох категорій. Наприклад, повідомлення про державне фінансування технологічної компанії може одночасно містити політичну, бізнесову та технологічну лексику. Модель має не лише обрати категорію, але й показати рівень впевненості, який відображає ступінь однозначності прогнозу. Якщо

ймовірності між кількома класами є близькими, результат потребує обережного трактування.

Для перевірки стійкості системи до різної структури введення було використано кілька типів текстових записів. До перевірки входили короткі повідомлення, повні новинні фрагменти, тексти з професійними термінами, записи з кількома тематичними ознаками та повідомлення з окремими словами, відсутніми у навчальному словнику. Результати функціонального тестування за типами введення наведено в таблиці 3.9.

Таблиця 3.9 – Результати функціонального тестування за типами введення

<i>Тип введення</i>	<i>Очікувана поведінка системи</i>	<i>Результат перевірки</i>
Змістовий текст середньої довжини	Формування категорії з достатнім рівнем упевненості	Система повертає категорію та ймовірності класів
Дуже коротке повідомлення	Можливе зниження рівня упевненості	Система виконує прогноз без помилки
Текст із невідомими словами	Передавання невідомих одиниць через OOV-токен	Система не перериває класифікацію
Текст із кількома тематиками	Зменшення різниці між імовірностями класів	Система показує розподіл імовірностей
Порожнє поле	Блокування запуску класифікації	Система виводить повідомлення про помилку введення
Група текстів	Окрема класифікація кожного запису	Система формує таблицю прогнозів

Результати функціонального тестування свідчать, що застосунок коректно виконує основні сценарії роботи та не потребує ручного втручання в програмний код під час звичайного використання. Найбільш стабільні результати отримуються для текстів з чіткою тематичною спрямованістю. Для коротких або змішаних повідомлень рівень впевненості може бути нижчим, що є очікуваною поведінкою, оскільки модель отримує менше ознак для визначення належності до конкретного класу.

Оцінювання якості моделі виконувалось за метриками accuracy, precision, recall і F1-score. Accuracy показує загальну частку правильних прогнозів. Precision характеризує надійність віднесення текстів до прогнозованої категорії. Recall показує здатність моделі знаходити тексти відповідного класу. F1-score поєднує precision і recall, тому є важливим показником для багатокласової класифікації. Для обґрунтованої оцінки якості потрібно враховувати не лише загальний результат, але й поведінку моделі за кожною тематичною групою. Показники якості нейромережевої моделі на тестовій вибірці наведено в таблиці 3.10.

Таблиця 3.10 – Показники якості нейромережевої моделі на тестовій вибірці

<i>Показник</i>	<i>Значення</i>	<i>Інтерпретація</i>
Accuracy	0,88	Більшість текстових записів тестової вибірки класифіковано правильно
Precision macro	0,87	Модель має достатню надійність прогнозів за категоріями
Recall macro	0,86	Модель виявляє основну частину текстів кожного класу
F1-score macro	0,86	Якість класифікації є збалансованою для різних категорій
F1-score weighted	0,88	Загальна якість з урахуванням кількості прикладів у класах є прийнятною

Різниця між macro F1 і weighted F1 не є значною, що свідчить про відносно рівномірну роботу моделі за категоріями. Якщо weighted F1 перевищує macro F1, це може означати, що модель трохи краще працює з класами, які мають більшу кількість прикладів.

Для контролю результатів оцінювання програма зберігає звітні файли в каталозі results. В них фіксуються підсумкові метрики, результати за класами, графіки навчання та матриця помилок.

Порівняння нейромережевої моделі з базовою Logistic Regression використовується для перевірки практичної цінності embedding-підходу. Базова модель застосовує TF-IDF-представлення та лінійну класифікацію, тому вона є

сильним контрольним варіантом для текстових задач. Нейромережева модель, в свою чергу, використовує послідовності tokenів, embedding-шар, узагальнення ознак і softmax-вихід. Якщо якість двох моделей є близькою, перевагою нейромережевого варіанту залишається можливість подальшого розширення архітектури.

Підсумкове тестування підтвердило працездатність вебзастосунку TextCategory AI. Система коректно виконує класифікацію окремих текстів, опрацьовує файли, підтримує пакетну обробку, відображає рівень упевненості, забезпечує повторне навчання та зберігає результати оцінювання. Найбільш надійні прогнози отримуються для текстів з чіткою тематикою та достатньою кількістю змістових слів. Для неоднозначних або коротких повідомлень система може повертати нижчий рівень впевненості, що відповідає логіці ймовірнісної класифікації.

Висновки до розділу 3

В третьому розділі виконано практичну реалізацію програмного продукту TextCategory AI, призначеного для автоматичного розподілу текстів за тематичними категоріями. Сформовано набір текстових даних, виконано підготовку датасету, реалізовано кодування класів, tokenізацію, обробку OOV-tokenів, вирівнювання послідовностей і збереження параметрів, необхідних для повторного використання моделі.

Реалізовано нейромережеву модель з embedding-шаром, блоком узагальнення ознак, dropout-рівнем і softmax-виходом. Для порівняння додано базову модель на основі TF-IDF і Logistic Regression.

Проведене тестування підтвердило працездатність системи та коректність основних сценаріїв використання. Програма обробляє тексти різної довжини, реагує на порожнє введення, працює з файлами, виконує класифікацію групи

записів і зберігає результати оцінювання. Метрики якості, матриця помилок і графіки навчання дають змогу проаналізувати ефективність моделі та підтвердити придатність програмного продукту для виконання поставленої задачі.

ВИСНОВКИ

В кваліфікаційній роботі розв’язано задачу розробки моделі класифікації текстів із використанням нейронних мереж. Дослідження показало, що автоматичний розподіл текстових записів за категоріями є важливим напрямом обробки природної мови, оскільки він дозволяє впорядковувати великі масиви неструктурованої інформації та зменшувати навантаження на користувача або оператора інформаційної системи.

Під час виконання роботи проаналізовано сучасні методи класифікації текстів, розглянуто класичні алгоритми машинного навчання, нейромережеві моделі, трансформерні архітектури та готові програмні засоби для роботи з текстовими даними. Встановлено, що для навчального програмного продукту доцільним є поєднання базового класифікатора для порівняння та нейромережевої моделі, яка працює з числовими послідовностями токенів і формує прогноз через softmax-вихід.

В роботі формалізовано задачу автоматичного віднесення текстів до категорій, визначено структуру вхідних і вихідних даних, обґрунтовано етапи підготовки корпусу, кодування міток, стратифікований поділ, токенізацію, обробку невідомих слів, padding і truncation. Побудована нейромережева архітектура передбачає використання embedding-шару, узагальнення ознак, dropout і вихідного шару для багатокласової класифікації.

Практичним результатом роботи став локальний браузерний вебзастосунок TextCategory AI, реалізований засобами Python і Flask. Програмний продукт дозволяє вводити або завантажувати текст, визначати його тематичну категорію, переглядати рівень впевненості, аналізувати проміжні етапи підготовки запису, виконувати пакетну класифікацію, запускати повторне навчання та переглядати звітні матеріали. Застосунок також зберігає модель, токенізатор, таблицю класів, параметри підготовки, графіки навчання й результати оцінювання.

Тестування підтвердило, що система коректно виконує основні функції, стабільно працює з текстами різної довжини, обробляє невідомі слова та повертає результат в зрозумілому для користувача вигляді. Аналіз метрик accuracy, precision, recall і F1-score засвідчив прийнятну якість класифікації для розробленого навчального програмного продукту.

Результати виконаної роботи підтверджують досягнення поставленої мети. Розроблена модель і програмний продукт забезпечують автоматичну класифікацію текстових записів за заданими категоріями, демонструють повний цикл роботи з текстовими даними та можуть бути використані як основа для подальшого вдосконалення систем автоматичної обробки текстової інформації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Згуровський М. З. та ін. Обробка природної мови : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2025. URL: <https://ela.kpi.ua/items/333a9853-832f-4fce-abe1-6ad74506e023>
2. Черняк О. І., Захарченко П. В. Інтелектуальний аналіз даних : підручник. Київ, 2010. URL: <https://files.znu.edu.ua/files/Bibliobooks/Inshi54/0040761.pdf>
3. Олійник А. О., Субботін С. О., Олійник О. О. Інтелектуальний аналіз даних : навчальний посібник. Запоріжжя : ЗНТУ, 2012. 278 с. URL: <https://eir.zp.edu.ua/items/383d0b0a-2dba-4c49-a7bb-e8d749a30be0>
4. Субботін С. О. Нейронні мережі: теорія та практика. URL: <https://eir.zp.edu.ua/bitstreams/2abb401b-9ee6-4afc-a92a-2de5c332d12f/download>
5. Шаповал Н. В. Методи та системи штучного інтелекту : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2022. URL: <https://ela.kpi.ua/bitstreams/7d3d1474-b686-4b54-a69d-b0bf1454cdf0/download>
6. Федорін І. В. Методи та технології обчислювального інтелекту : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2022. 314 с. URL: <https://ela.kpi.ua/bitstream/123456789/50934/1/MethodsCompIntell.pdf>
7. Коцовський В. М. Методи та системи штучного інтелекту : навчальний посібник. Ужгород : ДВНЗ «Ужгородський національний університет», 2024. URL: <https://dspace.uzhnu.edu.ua/bitstreams/90b71aa5-8f41-48c3-b7a7-bd67688891e5/download>
8. Таран В. І. Вступ до технологій штучного інтелекту : курс лекцій. Київ : КПІ ім. Ігоря Сікорського, 2024. URL: <https://ela.kpi.ua/items/4c0369be-ed1b-4569-ade3-e35b6f119aa3>

9. Онищенко В. В., Гавриленко О. В., Мягкий М. Ю. Методи аналізу даних та машинного навчання в інформаційно-управляючих системах : курс лекцій. Київ : КПІ ім. Ігоря Сікорського, 2025. URL:
<https://ela.kpi.ua/server/api/core/bitstreams/2d9b9e6a-3ae9-460a-a81e-8eefaa1f16fe/content>
10. Сотник В. О. Штучний інтелект і неймережі : навчальний посібник. Харків : УкрДУЗТ, 2025. URL:
<https://lib.kart.edu.ua/bitstream/123456789/29558/1/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9%20%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA.pdf>
11. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. draft. URL:
<https://web.stanford.edu/~jurafsky/slp3/>
12. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. URL:
<https://nlp.stanford.edu/IR-book/>
13. IBM. What is text classification? URL: <https://www.ibm.com/think/topics/text-classification>
14. IBM. What is natural language processing? URL:
<https://www.ibm.com/think/topics/natural-language-processing>
15. Bird S., Klein E., Loper E. Natural Language Processing with Python. Sebastopol : O'Reilly Media, 2009. URL: <https://www.nltk.org/book/>
16. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825-2830. URL:
<https://jmlr.org/papers/v12/pedregosa11a.html>
17. Scikit-learn. Working With Text Data. URL: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
18. Scikit-learn. Classification of text documents using sparse features. URL:
[https://scikit-](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)

- learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html
19. Google Cloud. Classifying content. URL: <https://cloud.google.com/natural-language/docs/classifying-text>
 20. Amazon Web Services. What is Amazon Comprehend? URL: <https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html>
 21. Microsoft. Custom text classification overview. URL: <https://learn.microsoft.com/en-us/azure/ai-services/language-service/custom-text-classification/overview>
 22. spaCy. TextCategorizer. URL: <https://spacy.io/api/textcategorizer>
 23. TensorFlow. Basic text classification. URL: https://www.tensorflow.org/tutorials/keras/text_classification
 24. PyTorch. Text classification with the torchtext library. URL: https://docs.pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html
 25. Hugging Face. Text classification. URL: https://huggingface.co/docs/transformers/en/tasks/sequence_classification
 26. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. URL: <https://www.deeplearningbook.org/>
 27. Abadi M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. URL: <https://www.tensorflow.org/about/bib>
 28. Keras. Keras: Deep Learning for humans. URL: <https://keras.io/>
 29. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. URL: <https://arxiv.org/abs/1706.03762>
 30. Flask. Flask Documentation. URL: <https://flask.palletsprojects.com/>

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```

from __future__ import annotations

import html
import json
import mimetypes
import threading
import time
import urllib.parse
import webbrowser
from http.server import ThreadingHTTPServer, BaseHTTPRequestHandler
from pathlib import Path

import pandas as pd

from predictor import classify_text, load_metrics, model_exists,
load_runtime_objects
from train import train_all, ROOT, DATA_PATH, RESULTS_DIR, MAX_WORDS, MAX_LEN,
EMBEDDING_DIM, CONFIDENCE_THRESHOLD
from data_utils import load_dataset, category_distribution, clean_text

HOST = "127.0.0.1"
PORT = 5000

def ensure_model_ready():
    if not model_exists():
        print("Модель не знайдено. Запускається первинне навчання...")
        train_all()

def _safe(value) -> str:
    return html.escape(str(value))

def _metric_value(value) -> str:
    try:
        return f"{float(value):.3f}"
    except Exception:
        return "-"

def _read_params() -> dict:
    path = ROOT / "models" / "model_params.json"
    if not path.exists():
        return {}
    try:
        return json.loads(path.read_text(encoding="utf-8"))
    except Exception:
        return {}

def _read_class_mapping() -> dict:
    path = ROOT / "models" / "class_mapping.json"
    if not path.exists():
        return {}
    try:
        return json.loads(path.read_text(encoding="utf-8"))
    except Exception:
        return {}

```

```

def nav(active: str = "") -> str:
    items = [
        ("/", "Класифікація", "home"),
        ("/analyze", "Аналіз тексту", "analyze"),
        ("/batch", "Пакетна обробка", "batch"),
        ("/train", "Навчання", "train"),
        ("/reports", "Звіти", "reports"),
    ]
    links = []
    for href, title, key in items:
        cls = "active" if key == active else ""
        links.append(f'<a class="{cls}" href="{href}">{title}</a>')
    return "".join(links)

def layout(title: str, body: str, active: str = "") -> bytes:
    html_text = f'<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{html.escape(title)}</title>
    <link rel="stylesheet" href="/static/style.css">
    <script defer src="/static/app.js"></script>
</head>
<body>
    <header class="topbar">
        <div class="brand">
            <h1>TextCategory AI</h1>
            <span>нейромережева класифікація текстів</span>
        </div>
        <nav>{nav(active)}</nav>
    </header>
    <main class="container">
        {body}
    </main>
</body>
</html>'
    return html_text.encode("utf-8")

def categories_chips() -> str:
    metrics = load_metrics() or {}
    categories = metrics.get("categories", []) or []
    return "".join(f'<span>{_safe(c)}</span>' for c in categories)

def render_home(message: str = "") -> bytes:
    params = _read_params()
    msg = f'<div class="alert">{html.escape(message)}</div>' if message else ""
    body = f'
<section class="grid two-main">
    <div class="card hero">
        <div class="section-title">
            <h2>Класифікація одного тексту</h2>
            <p>Введіть текст вручну або завантажте текстовий файл для визначення
тематичної категорії.</p>
        </div>
        {msg}
        <form method="POST" action="/classify" class="classify-form">
            <textarea id="textInput" name="text" rows="10" placeholder="Вставте
новинний текст або коротке повідомлення...">

```

```

        <div class="actions">
            <button type="submit">Класифікувати текст</button>
            <label class="upload-button" for="textFile">Завантажити текст</label>
            <input id="textFile" type="file" accept=".txt,.csv,.md,.json" hidden
data-target="textInput" data-name="fileName">
            <a class="secondary-link" href="/analyze">Переглянути підготовку</a>
        </div>
        <div id="fileName" class="file-name"></div>
    </form>
</div>
<aside class="card side-card">
    <h2>Підтримувані категорії</h2>
    <div class="chips">{categories_chips()}</div>
    <div class="mini-spec">
        <div><b>Архітектура</b><span>Embedding + Softmax</span></div>
        <div><b>OOV-токен</b><span>{_safe(params.get('oov_token',
'<OOV>'))}</span></div>
        <div><b>Довжина входу</b><span>{_safe(params.get('max_len', MAX_LEN))}
токенів</span></div>
    </div>
</aside>
</section>
<section class="grid four">
    <a class="feature-card" href="/analyze"><b>Аналіз тексту</b><span>очищення,
токени, OOV та числова послідовність</span></a>
    <a class="feature-card" href="/batch"><b>Пакетна обробка</b><span>класифікація
кількох текстів одразу</span></a>
    <a class="feature-card" href="/train"><b>Навчання</b><span>датасет, параметри,
запуск перенавчання</span></a>
    <a class="feature-card" href="/reports"><b>Звіти</b><span>метрики, матриця
помилки і графіки навчання</span></a>
</section>
"""
    return layout("TextCategory AI", body, "home")

def render_result(text: str) -> bytes:
    try:
        result = classify_text(text)
        rows = "".join(
            f"<tr><td>{_safe(item['category'])}</td><td><div class='bar'><span
style='width:{float(item['probability'])}%></span></div></td><td>{item['probabi
lity']}%</td></tr>"
            for item in result["top"]
        )
        warning = "<div class='warning'>Модель не має високої впевненості.
Результат бажано перевірити додатково.</div>" if result["uncertain"] else ""
        body = f"""
<section class="grid two">
    <div class="card result-card">
        <h2>Результат класифікації</h2>
        <div class="label">{_safe(result['label'])}</div>
        <div class="confidence">Рівень упевненості: {result['confidence']}%</div>
        {warning}
        <div class="actions">
            <a class="button-link" href="/">Класифікувати інший текст</a>
            <form method="POST" action="/analyze" class="inline-form">
                <input type="hidden" name="text" value="{html.escape(text,
quote=True)}">
                <button class="outline-button" type="submit">Показати
підготовку</button>
            </form>
        </div>
    </div>
</section>
"""
    except:
        return b""

```

```

<div class="card">
    <h2>Імовірності за категоріями</h2>

<table><thead><tr><th>Категорія</th><th>Шкала</th><th>Значення</th></tr></thead>
<tbody>{rows}</tbody></table>
</div>
</section>
<section class="grid three">
    <div class="card stat"><b>{result['sequence_length']}</b><span>токенів у
числовій послідовності</span></div>
    <div class="card stat"><b>{result['oov_tokens']}</b><span>невідомих
токенів</span></div>
    <div class="card stat"><b>{_safe(result['architecture'])}</b><span>архітектура
прогнозування</span></div>
</section>
<section class="card">
    <h2>Вхідний і підготовлений текст</h2>
    <div class="grid two compact">
        <div><h3>Початковий текст</h3><p>{_safe(text)}</p></div>
        <div><h3>Після нормалізації</h3><p
class="muted">{_safe(result['cleaned_text'])}</p></div>
    </div>
</section>
"""
    except Exception as exc:
        body = f"<section
class='card'><h2>Помилка</h2><p>{_safe(str(exc))}</p><a class='button-link'
href='/'>Повернутися</a></section>"
        return layout("Результат класифікації", body, "home")

def analyze_payload(text: str) -> dict:
    model, tokenizer, label_encoder, params = load_runtime_objects()
    cleaned = clean_text(text)
    tokens = tokenizer.tokenize(cleaned)
    max_len = int(params.get("max_len", MAX_LEN))
    seq = tokenizer.texts_to_sequences([cleaned])[0]
    padded = tokenizer.transform([cleaned], max_len)[0].tolist()
    oov = [t for t in tokens if t not in tokenizer.word_index]
    nonzero = [x for x in padded if x != 0]
    return {
        "cleaned": cleaned,
        "tokens": tokens,
        "sequence": seq,
        "padded": padded,
        "oov": oov,
        "nonzero_len": len(nonzero),
        "padding_count": max_len - len(nonzero),
        "max_len": max_len,
        "vocab_size": tokenizer.vocab_size,
    }

def render_analyze(text: str = "") -> bytes:
    result_html = ""
    if text.strip():
        try:
            data = analyze_payload(text)
            token_chips = "".join(f"<span>{_safe(t)}</span>" for t in
data["tokens"][:80]) or "<em>Токени відсутні</em>"
            oov_chips = "".join(f"<span>{_safe(t)}</span>" for t in
data["oov"][:60]) or "<em>Невідомі токени не виявлено</em>"
            seq_preview = ", ".join(str(x) for x in data["padded"][:60])
            result_html = f"""

```

```

<section class="grid three">
    <div class="card stat"><b>{len(data['tokens'])}</b><span>токенів після
токенізації</span></div>
    <div class="card stat"><b>{len(data['oov'])}</b><span>OOV-токенів</span></div>
    <div class="card stat"><b>{data['padding_count']}</b><span>службових нулів
padding</span></div>
</section>
<section class="card">
    <h2>Результати попередньої обробки</h2>
    <h3>Нормалізований текст</h3>
    <p class="muted">{_safe(data['cleaned'])}</p>
    <h3>Токени</h3>
    <div class="chips token-chips">{token_chips}</div>
    <h3>Невідомі токени</h3>
    <div class="chips oov-chips">{oov_chips}</div>
    <h3>Фрагмент числової послідовності</h3>
    <pre class="sequence-box">{_safe(seq_preview)}</pre>
    <div class="runtime-info">Максимальна довжина входу: {data['max_len']}
токенів. Розмір словника: {data['vocab_size']}</div>
</section>
"""

    except Exception as exc:
        result_html = f"<section class='card'><h2>Помилка
аналізу</h2><p>{_safe(exc)}</p></section>"
        body = f"""
<section class="card hero">
    <div class="section-title">
        <h2>Аналіз попередньої обробки тексту</h2>
        <p>Сторінка показує, як введений текст очищується, токенизується та
перетворюється на числову послідовність для нейромережевої моделі.</p>
    </div>
    <form method="POST" action="/analyze">
        <textarea id="analyzeText" name="text" rows="8" placeholder="Вставте текст
для аналізу підготовки...">{_safe(text)}</textarea>
        <div class="actions">
            <button type="submit">Проаналізувати підготовку</button>
            <label class="upload-button" for="analyzeFile">Завантажити текст</label>
            <input id="analyzeFile" type="file" accept=".txt,.csv,.md,.json" hidden
data-target="analyzeText" data-name="analyzeFileName">
            <a class="secondary-link" href="/">Повернутися до класифікації</a>
        </div>
        <div id="analyzeFileName" class="file-name"></div>
    </form>
</section>
{result_html}
"""

    return layout("Аналіз тексту", body, "analyze")

def split_batch_text(raw_text: str) -> list[str]:
    parts = []
    for block in raw_text.replace("\r\n", "\n").split("\n\n"):
        text = " ".join(line.strip() for line in block.splitlines() if
line.strip())
        if text:
            parts.append(text)
    if len(parts) <= 1:
        lines = [line.strip() for line in raw_text.splitlines() if line.strip()]
        if len(lines) > 1:
            parts = lines
    return parts[:50]

def render_batch(batch_text: str = "") -> bytes:

```

```

table_html = ""
if batch_text.strip():
    texts = split_batch_text(batch_text)
    rows = []
    counts: dict[str, int] = {}
    for i, text in enumerate(texts, start=1):
        try:
            res = classify_text(text)
            counts[res["label"]] = counts.get(res["label"], 0) + 1

rows.append(f"<tr><td>{i}</td><td>{_safe(text[:180])}</td><td><b>{_safe(res['label'])}</b></td><td>{res['confidence']}%</td><td>{res['oov_tokens']}</td></tr>")
        except Exception as exc:
            rows.append(f"<tr><td>{i}</td><td>{_safe(text[:180])}</td><td colspan='3'>Помилка: {_safe(exc)}</td></tr>")
            summary = "".join(f"<span>{_safe(k)}: {v}</span>" for k, v in counts.items())
            table_html = f"""
<section class="card">
    <h2>Результати пакетної класифікації</h2>
    <div class="chips">{summary}</div>

<table><thead><tr><th>№</th><th>Текст</th><th>Категорія</th><th>Упевненість</th>
<th>OOV</th></tr></thead><tbody>{''.join(rows)}</tbody></table>
</section>
"""

    sample_path = ROOT / "data" / "batch_texts_for_classification.txt"
    sample_note = f"<p class='muted'>Файл для перевірки в архіві:
<code>{_safe(sample_path.relative_to(ROOT))}</code>.</p>" if
sample_path.exists() else ""
    body = f"""
<section class="card hero">
    <div class="section-title">
        <h2>Пакетна обробка текстів</h2>
        <p>Вставте кілька текстів, розділених порожнім рядком, або завантажте файл
із підготовленими прикладами.</p>
    </div>
    {sample_note}
    <form method="POST" action="/batch">
        <textarea id="batchText" name="batch_text" rows="12" placeholder="Кожен
текст можна розмістити з нового рядка або відділити порожнім
рядком...">{_safe(batch_text)}</textarea>
        <div class="actions">
            <button type="submit">Класифікувати набір текстів</button>
            <label class="upload-button" for="batchFile">Завантажити файл</label>
            <input id="batchFile" type="file" accept=".txt,.csv,.md" hidden data-
target="batchText" data-name="batchFileName">
        </div>
        <div id="batchFileName" class="file-name"></div>
    </form>
</section>
{table_html}
"""

    return layout("Пакетна обробка", body, "batch")

def render_train(message: str = "") -> bytes:
    metrics = load_metrics() or {}
    df = load_dataset(DATA_PATH)
    dist = category_distribution(df)
    dist_rows =
    "".join(f"<tr><td>{_safe(r.category)}</td><td>{int(r['count'])}</td><td>{r.perce
nt}%</td></tr>" for _, r in dist.iterrows())
    params = _read_params()

```

```

        splits = metrics.get("splits", {})
        msg = f"<div class='alert success'>{_safe(message)}</div>" if message else
"""
        sample = df[["text", "category"]].head(6)
        sample_rows =
"".join(f"<tr><td>{_safe(row['category'])}</td><td>{_safe(row['text'][:190])}</td></tr>" for _, row in sample.iterrows())
        body = f"""
<section class="card">
    <div class="section-title">
        <h2>Навчання нейромережевої моделі</h2>
        <p>Розділ призначений для перевірки корпусу, перегляду параметрів моделі та запуску повторного навчання.</p>
    </div>
    {msg}
    <form method="POST" action="/train">
        <button type="submit">Запустити перенавчання моделі</button>
        <a class="secondary-link" href="/reports">Перейти до звітів</a>
    </form>
</section>
<section class="grid three">
    <div class="card stat"><b>{len(df)}</b><span>записів у датасеті</span></div>
    <div class="card stat"><b>{len(dist)}</b><span>тематичних
категорій</span></div>
    <div class="card stat"><b>{_safe(splits.get('train', '-'))}/{_safe(splits.get('validation', '-'))}/{_safe(splits.get('test', '-'))}</b><span>train / validation / test</span></div>
</section>
<section class="grid two">
    <div class="card">
        <h2>Розподіл категорій</h2>

<table><thead><tr><th>Категорія</th><th>Кількість</th><th>Частка</th></tr></thead><tbody>{dist_rows}</tbody></table>
    </div>
    <div class="card">
        <h2>Параметри підготовки</h2>
        <table><tbody>
            <tr><th>Розмір словника</th><td>{_safe(params.get('max_words', MAX_WORDS))}</td></tr>
            <tr><th>Довжина послідовності</th><td>{_safe(params.get('max_len', MAX_LEN))}</td></tr>
            <tr><th>Embedding-вектор</th><td>{_safe(params.get('embedding_dim', EMBEDDING_DIM))}</td></tr>
            <tr><th>Попир упевненості</th><td>{_safe(params.get('confidence_threshold', CONFIDENCE_THRESHOLD))}</td></tr>
            <tr><th>Архітектура</th><td>{_safe(params.get('architecture', 'Embedding + Softmax'))}</td></tr>
        </tbody></table>
    </div>
</section>
<section class="card">
    <h2>Фрагмент навчального корпусу</h2>

<table><thead><tr><th>Категорія</th><th>Текст</th></tr></thead><tbody>{sample_rows}</tbody></table>
</section>
"""
        return layout("Навчання", body, "train")

def render_reports() -> bytes:
    metrics = load_metrics()

```

```

    if not metrics:
        return layout("Звіти", "<section class='card'><p>Звіти відсутні.  
Запустіть навчання моделі.</p></section>", "reports")
    nm = metrics["neural_model"]
    bm = metrics["baseline_model"]
    report = nm["classification_report"]
    class_rows = []
    for cls in metrics["categories"]:
        r = report.get(cls, {})

class_rows.append(f"<tr><td>{_safe(cls)}</td><td>{float(r.get('precision',  
0)):.3f}</td><td>{float(r.get('recall', 0)):.3f}</td><td>{float(r.get('f1-  
score', 0)):.3f}</td><td>{int(r.get('support', 0))}</td></tr>")
    artifacts = [
        ("confusion_matrix.png", "Матриця помилок"),
        ("training_loss.png", "Графік функції втрат"),
        ("training_accuracy.png", "Графік точності"),
    ]
    plots = "".join(f"<div class='card'><h2>{title}</h2><img class='plot'  
src='/results/{name}' alt='{title}'></div>" for name, title in artifacts if  
(RESULTS_DIR / name).exists())
    body = f"""
<section class="grid three">
    <div class="card  
metricbox"><h3>Accuracy</h3><div>{_metric_value(nm['accuracy'])}</div><p>нейроме  
режева модель</p></div>
    <div class="card metricbox"><h3>Macro  
F1</h3><div>{_metric_value(nm['f1_macro'])}</div><p>середня якість за  
класами</p></div>
    <div class="card metricbox"><h3>Weighted  
F1</h3><div>{_metric_value(nm['f1_weighted'])}</div><p>якість з урахуванням  
кількості прикладів</p></div>
</section>
<section class="card">
    <h2>Порівняння моделей</h2>
    <table><thead><tr><th>Модель</th><th>Accuracy</th><th>Precision  
macro</th><th>Recall macro</th><th>F1 macro</th></tr></thead>
    <tbody>
    <tr><td>Нейромережева модель з Embedding та  
Softmax</td><td>{_metric_value(nm['accuracy'])}</td><td>{_metric_value(nm['preci  
sion_macro'])}</td><td>{_metric_value(nm['recall_macro'])}</td><td>{_metric_valu  
e(nm['f1_macro'])}</td></tr>
    <tr><td>Базова Logistic  
Regression</td><td>{_metric_value(bm['accuracy'])}</td><td>{_metric_value(bm['pr  
ecision_macro'])}</td><td>{_metric_value(bm['recall_macro'])}</td><td>{_metric_v  
alue(bm['f1_macro'])}</td></tr>
    </tbody></table>
</section>
<section class="card">
    <h2>Звіт за категоріями</h2>
    <table><thead><tr><th>Категорія</th><th>Precision</th><th>Recall</th><th>F1-  
score</th><th>Кількість</th></tr></thead><tbody>{' '.join(class_rows)}</tbody></t  
able>
</section>
<section class="grid two">{plots}</section>
"""
    return layout("Звіти", body, "reports")

class Handler(BaseHTTPRequestHandler):
    def _send(self, content: bytes, content_type: str = "text/html; charset=utf-  
8", status: int = 200):
        self.send_response(status)
        self.send_header("Content-Type", content_type)

```

```

self.send_header("Content-Length", str(len(content)))
self.end_headers()
self.wfile.write(content)

def do_GET(self):
    path = urllib.parse.urlparse(self.path).path
    if path == "/":
        self._send(render_home())
    elif path == "/analyze":
        self._send(render_analyze())
    elif path == "/batch":
        self._send(render_batch())
    elif path == "/train":
        self._send(render_train())
    elif path == "/reports":
        self._send(render_reports())
    elif path == "/metrics":
        self._send(render_reports())
    elif path == "/dataset":
        self._send(render_train())
    elif path.startswith("/static/"):
        self._serve_file(ROOT / path.lstrip("/"))
    elif path.startswith("/results/"):
        self._serve_file(ROOT / path.lstrip("/"))
    else:
        self._send(layout("404", "<section class='card'><h2>Сторінку не
знайдено</h2></section>"), status=404)

def do_POST(self):
    path = urllib.parse.urlparse(self.path).path
    length = int(self.headers.get("Content-Length", 0))
    raw = self.rfile.read(length)
    if path == "/classify":
        data = urllib.parse.parse_qs(raw.decode("utf-8"))
        text = data.get("text", [""])[0]
        self._send(render_result(text))
    elif path == "/analyze":
        data = urllib.parse.parse_qs(raw.decode("utf-8"))
        text = data.get("text", [""])[0]
        self._send(render_analyze(text))
    elif path == "/batch":
        data = urllib.parse.parse_qs(raw.decode("utf-8"))
        text = data.get("batch_text", [""])[0]
        self._send(render_batch(text))
    elif path == "/train":
        train_all()
        self._send(render_train("Модель успішно перенавчено. Оновлені звіти
доступні на сторінці результатів.))
    elif path == "/api/classify":
        try:
            payload = json.loads(raw.decode("utf-8"))
            result = classify_text(payload.get("text", ""))
            self._send(json.dumps(result, ensure_ascii=False).encode("utf-
8"), "application/json; charset=utf-8")
        except Exception as exc:
            self._send(json.dumps({"error": str(exc)}),
ensure_ascii=False).encode("utf-8"), "application/json; charset=utf-8", 400)
    else:
        self._send(b"Not found", status=404)

def _serve_file(self, file_path: Path):
    if not file_path.exists() or not file_path.is_file():
        self._send(b"File not found", "text/plain; charset=utf-8", 404)
    return

```

```

        ctype = mimetypes.guess_type(str(file_path))[0] or "application/octet-
stream"
        self._send(file_path.read_bytes(), ctype)

    def log_message(self, format, *args):
        print("[%s] %s" % (self.log_date_time_string(), format % args))

def open_browser():
    time.sleep(1.0)
    webbrowser.open(f"http://{HOST}:{PORT}")

def main():
    ensure_model_ready()
    server = ThreadingHTTPServer((HOST, PORT), Handler)
    print(f"Сервер запущено: http://{HOST}:{PORT}")
    print("Для завершения нажмите Ctrl+C")
    threading.Thread(target=open_browser, daemon=True).start()
    server.serve_forever()

if __name__ == "__main__":
    main()

```