Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра мережних технологій

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**АВТОМАТИЧНА КЛАСИФІКАЦІЯ ТЕКСТІВ**»

Виконав: студент 4-го року навчання,
Освітньої програми «Інженерія програмного забезпечення», 121

Дубовик Андрій Вікторович

Керівник Волинець Є.А.,_____
ст. викладач, к.н.

Рецензент _____
(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2024

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мережних технологій

ЗАТВЕРДЖУЮ

Зав. кафедри мережних технологій

проф., д.ф.-м.н.

_____ Г. І. Малашонок

(підпис)

“____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Дубовику Андрію Вікторовичу 4-го курсу факультету інформатики

ТЕМА Автоматична класифікація текстів

Зміст ТЧ до кваліфікаційної роботи:

Вступ

Розділ 1. Аналіз предметної області

Розділ 2. Алгоритм розв’язку

Розділ 3. Реалізація системи класифікації

Висновки

Список використаних джерел

Дата видачі „___” _____ 2024 р. Волинець Є.А. _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи

Тема: Автоматична класифікація текстів

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на кваліфікаційну роботу	листопад 2023 р.	
2.	Огляд літератури за темою роботи	листопад-грудень 2023 р.	
3.	Розробка алгоритмів класифікації текстів	січень-березень 2024 р.	
4.	Написання текстової частини роботи	квітень 2024 р.	
5.	Коригування виконаної роботи	травень 2024 р.	
6.	Створення слайдів для доповіді та написання доповіді	травень 2024 р.	
7.	Подання роботи на кафедру для перевірки на плагіат	травень 2024 р.	
8.	Захист кваліфікаційної роботи	травень 2024 р.	

Студент Дубовик А. В.

Керівник Волинець Є. А.

“_____” _____

Зміст

Анотація

У цій роботі розглядаються різні підходи до класифікації текстів. Досліджено та порівняно різні методи класифікації. Результати програмних реалізацій порівнюються та аналізуються за точністю класифікації, часом виконання та іншими метриками ефективності. Висновки цього дослідження допоможуть визначити найбільш ефективні підходи до класифікації текстів та їх можливості в різних областях. У роботі детально описано розробку застосунку, який використовує комбінацію декількох методів класифікації.

Перелік термінів та умовних позначень

Дата сет (англ. dataset) – це збірка необроблених даних, що зазвичай застосовується в задачах машинного навчання чи обробки даних [1].

Матриця невідповідностей (англ. confusion matrix) – таблиця, яка підсумовує кількість правильних і неправильних передбачень, зроблених моделлю класифікації, кожен з рядків цієї таблиці представляє зразки прогнозованого класу, а кожен зі стовпців представляє зразки справжнього класу (або навпаки), може допомогти визначати паттерни помилок [1].

Трансформер – це архітектура нейронної мережі заснована на механізмі уваги, , розроблена Google [1].

BERT (Bidirectional Encoder Representations from Transformers) – архітектура моделі для представлення тексту, що ґрунтується на трансформері, розроблена Google [1].

RNN (Recurrent Neural Network) – рекурентна нейронна мережа; нейронна мережа, яка навмисно запускається кілька разів, де частини кожного циклу переходять у наступний цикл [1].

LSTM (Long short-term memory) – тип RNN, спрямований на розв’язування проблеми зникання градієнта, присутньої в традиційних рекурентних нейронних мережах [1].

SVM (Support Vector Machine) – це алгоритм машинного навчання, який використовується для класифікації або регресії, шляхом пошуку гіперплощини, що найкраще розділяє дані у просторі ознак [1].

LLM (Large Language Model) – термін без чіткого визначення, який зазвичай означає мовну модель, яка має велику кількість параметрів [1].

ВСТУП

Обробка текстової інформації має суттєве значення в різних сферах знань сучасного світу, таких як бізнес, наука, соціальні медіа та інші. З розвитком інформаційних технологій та зростанням обсягу доступної інформації стає надзвичайно важливим ефективне управління та аналіз цими даними. Однією з ключових задач в цьому контексті є автоматична класифікація текстів, яка полягає в розподілі текстів за певними категоріями не вдаючись при цьому до ручного аналізу. Під “категорією” може матися на увазі тема, почуття, мова тексту тощо. Це може бути важливим у багатьох установах для створення архівів і організації великих обсягів документів. Ця тема стає дедалі актуальнішою з розвитком Інтернету, де величезні обсяги різноманітної інформації потребують швидкого та точного аналізу. Зокрема, інформаційні системи, що використовують технології машинного навчання, такі як нейронні мережі та методи обробки природної мови (NLP), демонструють високу ефективність у розв'язанні цієї задачі завдяки своїй здатності адаптуватися до нових даних та швидкості обробки. Класифікація тексту може бути застосована в таких процесах:

- фільтрація спаму, процес, який намагається відрізнити спам-повідомлення з електронної пошти користувача (або з будь-якого іншого ресурсу) від реальних змістовних повідомлень;
- розпізнавання мови;
- ідентифікація жанру;
- категоризація новин;
- аналіз настроїв, визначення ставлення автора до певної теми;
- тощо [2] [3].

Основні виклики, пов'язані з автоматичною класифікацією текстів, це різноманітність форматів, структур текстів та семантична складність мови. Для вирішення цих проблем потрібні ефективні алгоритми та методи обробки природної мови.

Мета цієї роботи - аналіз та порівняння різних методів та технологій автоматичної класифікації текстів. Робота також охоплює розробку програмних засобів, що базуються на різних методах машинного навчання та обробки природної мови.

Практична частина роботи написана мовою програмування Python. Для розробки використовувалась IDE PyCharm та сервіс Google Colab. Для більшості функцій машинного навчання використано бібліотеки TensorFlow та scikit-learn. Для аналізу використано дата сет AG News з веб-сайту kaggle.com.

Текстова частина даної роботи складається з вступу, трьох основних розділів та висновків.

У першому розділі “Аналіз предметної області” розглядаються основні поняття галузі, описуються основні підходи до розв’язання проблеми, а також їхні переваги та недоліки.

У другому розділі “Алгоритм розв’язку” наведено загальний план розв’язання проблеми, а також описано необхідні для цього інструменти.

У третьому розділі “Реалізація системи класифікації” описано процес розробки класифікатора тексту з використанням різних алгоритмів та створення графічного інтерфейсу.

У висновках вказані підсумки даної роботи та можливі перспективи подальших досліджень.

Всі посилання на джерела інформації, які були використані в цій роботі, знаходяться у списку використаних джерел.

РОЗДІЛ 1 : АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основні поняття

Класифікація є процесом розподілу об'єктів або даних на певні категорії або класи відповідно до їхніх характеристик або властивостей. В контексті обробки текстів, класифікація полягає у призначенні кожному текстовому документу або фрагменту відповідної категорії або класу, що допомагає в організації, розумінні та аналізі великих обсягів інформації [3].

До XX століття класифікація текстів здійснювалася виключно вручну. Наприклад, у середньовіччі, монахи та учені працювали над індексами та каталогами, які класифікували тексти за темою або автором. У XIX столітті, з поширенням бібліотек та друку, було створено системи класифікації книг, такі як Десяткова класифікація Дьюї та Класифікація Бібліотеки Конгресу.

Потреба в класифікації текстів стала ще більш актуальною з появою комп'ютерів. Комп'ютери дозволили автоматизувати процес обробки текстів та розробити алгоритми для автоматичної класифікації текстових даних. Перші комп'ютерні системи для класифікації текстів з'явилися ще в другій половині XX століття, але вони були дуже примітивними порівняно з сучасними системами. Технології штучного інтелекту, зокрема методи машинного навчання, дозволили значно покращити ефективність та точність класифікації текстів.

Автоматична класифікація тексту є одним із фундаментальних завдань обробки природної мови і її дослідження має важливе значення у багатьох сферах життя. Наприклад, у наукових дослідженнях відсутність ефективних засобів для аналізу великих обсягів наукових публікацій може ускладнити синтез знань та виявлення тенденцій у розвитку науки. У сфері бізнесу, автоматична класифікація текстів може допомогти в розподілі замовлень або в аналізі ринку, що сприяє прийняттю обґрунтованих управлінських рішень. Також, у сфері соціальних мереж та медіа,

класифікація текстових повідомлень може допомогти в аналізі громадської думки та виявленні тенденцій у суспільстві.

1.2. Підходи до класифікації текстів

Всі системи автоматичної класифікації текстів можна умовно поділити на такі три групи:

- системи на основі правил;
- системи на основі машинного навчання;
- гібридні системи.

1.2.1. Системи на основі правил

Одним з найпростіших та найбільш зрозумілих методів класифікації текстів є системи на основі правил. Ці системи базуються на використанні набору правил, які визначають спосіб прийняття рішень щодо класифікації текстових даних [4]. Прикладом може бути проста система фільтрації спаму електронної пошти, яка відносить повідомлення, які насичені словами “акція”, “знижка”, “продаж”, “виграш”, “лотерея”, до спаму. Хоча й цей підхід може спершу здатися досить обмеженим порівняно з більш складними методами машинного навчання, він все ще знаходить своє застосування у деяких випадках. Перевагою систем на основі правил є їх прозорість та зрозумілість: ви легко можете зрозуміти, які правила були застосовані до конкретного тексту для класифікації. Однак недоліком цих систем є їх обмежена гнучкість. Також, такі системи можуть бути відносно складними у побудові та підтримці, особливо при обробці великих обсягів текстових даних.

1.2.2. Системи на основі машинного навчання

Системи класифікації текстів на основі машинного навчання використовують різноманітні алгоритми та моделі для автоматичного визначення категорій або класів, до яких належать тексти [4]. Ці системи зазвичай навчаються на зразках текстів з уже відомими мітками категорій, щоб зрозуміти певні закономірності у вхідних даних та правильно класифікувати нові зразки. Одними з найбільш широко застосовуваних рішень для класифікації текстів є:

- Naive Bayes.
- SVM (Support Vector Machine).
- RNN (Recurrent Neural Network).
- BERT (Bidirectional Encoder Representations from Transformers).
- LLM (Large Language Models).

1.2.2.1. Naive Bayes

Naive Bayes - це проста техніка для побудови класифікаторів, що ґрунтується на теоремі Баєса та припущенні про незалежність між ознаками [5]. Наприклад, якщо фрукт зелений або червоний, круглої форми й має радіус близько п’яти сантиметрів, то його можна вважати яблуком за допомогою наївного баєсіва класифікатора, який розглядає кожну ознаку як незалежну від інших при визначенні ймовірності того, що цей фрукт є яблуком. Перевагами такого методу є його ефективність для великих обсягів даних, простота в реалізації та швидкість. Недоліком є те, що припущення про незалежність між ознаками не завжди може бути справедливим і показувати гіршу ефективність порівняно з іншими методами.

Для текстової класифікації найчастіше використовується мультиноміальний баєсів класифікатор, в якому розподіл є параметризованим векторами $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ для кожного класу y, де n кількість ознак (в даному випадку розмір словника) та θ_{yi} це вірогідність $P(x_i|y)$ ознаки i з’явитись в екземплярі класу y. Параметри θ_{yi} оцінюються підрахунком відносної частоти:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

де $N_{yi} = \sum_{x \in T} x_i$ це кількість разів, скільки ознака i трапляється в класі y в тренувальному наборі даних T, і $N_y = \sum_{i=1}^{n} N_{yi}$ це загальна кількість всіх ознак в класі y. Попереднє згладжування $\alpha \geq 0$ враховує функції, яких немає у зразках навчання, і запобігає нульовій ймовірності в подальших обчисленнях. Встановлення $\alpha = 1$ називається згладжуванням Лапласа, в той час як $\alpha < 1$ називається згладжуванням Лідстоуна [5].

1.2.2.2. SVM

SVM – алгоритм, який шукає гіперплощину у просторі ознак, що найкращим чином розділяє екземпляри на різні категорії [1]. Перевагами такого методу є ефективність при роботі з розрідженими даними. Недоліками є те, що при великій кількості зразків для тренування або великій кількості ознак алгоритм може стати обчислювально-вимогливим, і також може демонструвати гірші, ніж інші алгоритми, результати при обробці великих корпусів текстів.

Загалом SVM працює таким чином. Ми маємо набір n точок з тренувального набору даних $(x_1, y_1), \dots (x_n, y_n)$, де y_i (1 або -1) вказує на належність точки x_i до певного класу. Кожен x_i – це p-розмірний вектор і SVM шукає гіперплощину, яка з максимальним запасом розділяє групу точок x_i для яких $y_1 = 1$, від тих точок для яких $y_1 = -1$.

Ілюстрацію роботи опорно-векторної машини можна побачити на рисунку нижче. H_1не розділяє задані класи, H_2 розділяє класи, проте не ідеально, а H_3 розділяє ідеально (з максимальним запасом) [6].

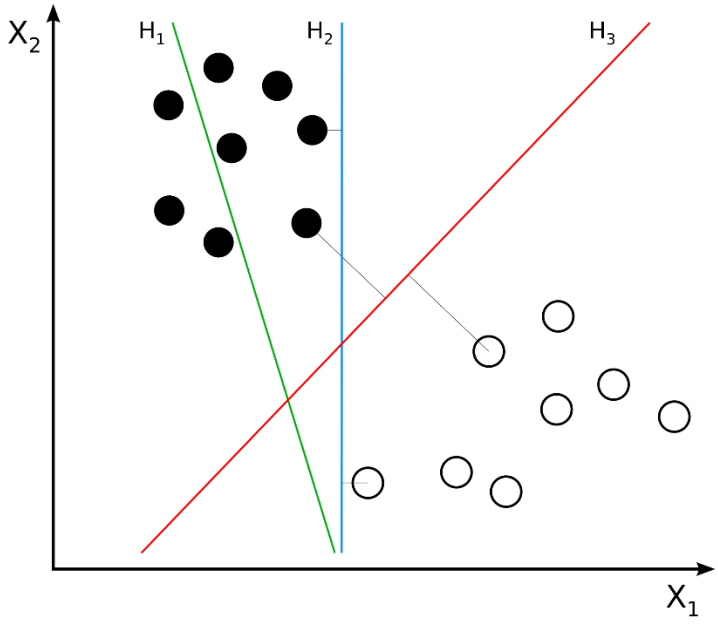

Рисунок 1.1. Ілюстрація роботи SVM [6]

1.2.2.2. RNN

RNN – рекурентна нейронна мережа, може ефективно працювати з послідовностями даних, такими як рядки тексту, та враховувати контекстуальні

зв'язки між словами [3]. Перевагою мереж такого типу є здатність до врахування контексту, що може бути корисним для визначення залежностей між словами та фразами в реченні. Проте обробка довгих текстових послідовностей для RNN може більш обчислювальна затратною операцією, ніж для Naive Bayes та SVM. Також рекурентні нейронні мережі мають проблему затухання градієнта і можуть погано зв’язувати інформацію при великій відстані між словами. Але як вихід з такої ситуації в 1997 році була запропонована модель LSTM, якій протягом останніх років вдалось набагато покращити можливості RNN. Наприклад повідомляється, що у 2015 році розпізнавання мовлення від Google значно покращилось завдяки використанню LSTM [7].

1.2.2.4. BERT

BERT - це одна з найпродуктивніших моделей у сфері обробки природної мови, що базується на трансформерах [8]. Модель зазвичай попередньо навчається на великому корпусі тексту, а потім підлаштовується для конкретних завдань. Основними перевагами BERT є здатність розуміти контексте представлення слів та його масштабованість. Недоліками ж є необхідність значних обчислювальних ресурсів для тренування та підлаштування, а також складність у налаштуванні порівняно з більш традиційними методами.

1.2.2.5. LLM

LLM - моделі, які базуються на нейронних мережах з великою кількістю параметрів та навчаються на великих обсягах текстових даних з метою розуміти та генерувати контент [9]. Перевагою таких моделей є їхня гнучкість, адже уже натреновані моделі такого типу здатні визначити дуже велику кількість категорій тексту. Проте значним недоліком моделей такого типу є необхідність потужної обчислювальної машини для роботи та необхідність надзвичайно великого обсягу даних для тренування у випадку, якщо ми створюємо власну модель, а не використовуємо наявну. Деякі відомі LLM – це моделі серії GPT від OpenAI (наприклад, GPT-3.5 і GPT-4, що використовуються в ChatGPT і Microsoft Copilot), PaLM і Gemini від Google (останній наразі використовується в однойменному чат-

боті), xAI Grok, сімейство моделей LLaMA від Meta. Але саме браузерний ChatGPT 2022 року, орієнтований на споживачів, захопив уяву широких верств населення та викликав ажіотаж у ЗМІ [10].

1.2.3. Гібридні системи

Гібридні системи класифікації текстів можуть поєднувати у собі переваги різних підходів та методів для створення більш точних та ефективних моделей [11]. Ці системи можуть використовувати комбінації всіх інших методів для досягнення кращих результатів. Вони становлять собою потужний інструмент, який може бути налаштований та оптимізований для конкретних потреб. Такі моделі можуть досягати високої точності та ефективності у різних сценаріях та областях застосування. Отже, гібридні системи класифікації текстів є ефективним інструментом, який можна використовувати для різноманітних завдань у багатьох галузях, від аналізу текстових даних до автоматичної обробки природної мови. Проте один з основних недоліків гібридних систем полягає в їх складності. Поєднання різних методів і підходів може призвести до значного збільшення складності самої системи, включаючи розробку, розуміння, підтримку та масштабування. Також використання комбінації різних методів може вимагати значних ресурсів для обчислення.

1.3 Постановка проблеми

Існує багато різних методів для класифікації текстів, кожен з яких має власні переваги та недоліки, щоб отримувати ефективні результати для різних корпусів текстів потрібно використовувати різні методи або їхню комбінацію.

Крім того, важливою функціональністю є можливість користувача навчати модель на власних текстових даних. Це відкриває широкі можливості для персоналізації та адаптації системи до конкретних потреб користувача.

Отже, потрібно розробити систему, яка зможе доволі швидко та точно класифікувати тексти. У користувача має бути можливість зручним способом натренувати систему на власних даних та налаштувати параметри для оптимальних

результатів. Система буде попередньо навчена на деякому наборі даних (в цьому випадку AG News Classification Dataset), що дозволить одразу розпізнавати деякі базові категорії текстів.

РОЗДІЛ 2 : АЛГОРИТМ РОЗВ'ЯЗКУ

2.1. Вибір інструментів

Для того, щоб ефективно опрацювати вхідні дані та реалізувати алгоритм класифікації необхідні відповідні інструменти. Тому треба визначити мову програмування, яка буде використовуватись для розробки та основні бібліотеки, які зможуть надати базовий необхідний функціонал.

В якості мови програмування використовуватиметься Python, оскільки саме для цієї мови існує сформована екосистема бібліотек для машинного навчання, обробки даних, матричної математики тощо. Синтаксис є лаконічним і зрозумілим, що в деяких випадках дозволяє реалізовувати більш складні алгоритми швидше та з меншими зусиллями. Великий вибір різноманітних бібліотек, простота та зручність використання, а також всі інші згадані фактори роблять Python стандартним вибором для проектів, що стосуються машинного навчання та нейронних мереж.

TensorFlow – безкоштовна бібліотека для машинного навчання з відкритим кодом [12]. Вона відома своєю масштабованістю та високою продуктивністю, що дозволяє легко створювати та навчати складні нейронні мережі.

scikit-learn – ще одна бібліотека для машинного навчання, яка надає простий та зрозумілий інтерфейс для реалізації класичних алгоритмів [13]. Вона містить великий набір інструментів для класифікації та інших завдань.

nltk (Natural Language Toolkit) – це бібліотека для опрацювання природної мови [14]. Надає широкий спектр інструментів для різноманітних завдань пов'язаних з обробкою природної мови.

Також нам знадобляться такі бібліотеки:

- NumPy [15] – для зручного доступу до великих та багатовимірних масивів, а також функцій високого рівня для роботи з такими масивами.
- Pandas [16] – для доступу до структури даних DataFrame, яка є швидкою, гнучкою та інтуїтивно зрозумілою.
- Matplotlib [17] та seaborn [18] – для візуалізації даних та побудови графіків.

Цей список засобів не є вичерпним, адже для деяких потреб також використовуватимуться й інші бібліотеки. Але саме описані в даному розділі інструменти відіграватимуть ключову роль при реалізації програмної частини.

2.2. Аналіз вхідних даних

Почати варто з аналізу вхідних даних. В даному випадку для навчання моделей буде використано "AG News Classification Dataset" з kaggle.com[19]. Всього даний

датасет містить 120 тисяч коротких текстів, зібраних з різних джерел новин. Всі тексти рівномірно (по 30 тисяч) розподілені між наступними чотирма категоріями:

- World (Світ) – тексти про події та новини з різних країн світу, міжнародні відносини, а також глобальні проблеми, такі як зміна клімату, міграція та інші.
- Sports (Спорт) – тексти про спортивні події, змагання, турніри, команди та виступи спортсменів у різних дисциплінах, таких як футбол, баскетбол, теніс та інші.
- Business (Бізнес) – тексти про компанії, фінансові новини, аналіз ринків, бізнес-стратегії, економічні тенденції та інші аспекти бізнесу, включаючи корпоративні злиття та поглиблені розгляди економічних подій різного роду.
- Science/Technology (Наука/Технології) – тексти про нові відкриття в науці, технологічні досягнення, інновації у технологічній сфері, включаючи роботи над штучним інтелектом, космічними відкриттями, медичними технологіями тощо.

Приклади текстів різних категорій з даного дата сету можна побачити нижче.

	label	text
58151	Sports	GOLF: SCOTT #39;S TO HAVE TEE WITH ERNIE. SCOTT DRUMMOND has been handed the task of taking on No.1 seed Ernie Els in the first round of the HSBC World Matchplay Championship at Wentworth on Thursday.
67709	Science/ Technology	Ebay Explodes to Twice Its Size. Executives at eBay apologized to frustrated customers who were frozen out of the online auction #39;s payment transaction service, which experienced at least five days of intermittent failures earlier this month.
17203	Sports	Liuzzi to test for Sauber. Sauber Petronas have invited Vitantonio Liuzzi for a test drive. The 22-year old Red Bull Junior Team pilot, who has won this year #39;s Formula 3000 Championship far ahead of the competition, will be driving the Sauber Petronas C23 at Jerez on 16 September.
22839	Science/ Technology	RealNetworks Sells 3 Mln Songs, Ends Download Promo (Reuters). Reuters - RealNetworks Inc. is ending\its 49 cent-per-song music download service but will keep the\promotional prices in place for top 10 songs, the Internet\media and software company said on Thursday.
31181	Science/ Technology	FTC Assesses Bounty System for Catching Spammers. The FTC today issued a report assessing whether and how a system that rewards members of the public for tracking down spammers would or could help improve enforcement of the Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2002
119436	World	Bug found at UN Geneva HQ. GENEVA, DECEMBER 17: The United Nations said on Thursday a secret listening device had been found in a posh meeting room of its European headquarters in Geneva.
71841	Business	NY Gold Ends Near \$430. NEW YORK (Reuters) - COMEX gold futures closed at fresh 6-1/2-month highs on Monday as investors bought the safe-haven metal due to an ailing dollar, soaring oil prices and uncertainty over the U.S. presidential election, dealers said.
71032	Science/ Technology	RFID cell phones take shape at Nokia. The cell phone maker is working on a phone that uses controversial microchips used to store product information and signal their location.
18882	World	India and Pakistan agree to widen peace dialogue . NEW DELHI -- Nuclear rivals India and Pakistan agreed yesterday to widen their peace dialogue in talks that focused on eight festering issues, including the decades-old dispute over the Himalayan region of Kashmir.
8522	World	Hicks to be reunited with father. ELEANOR HALL: After close to three years in US detention, Australian Guantanamo Bay detainee, David Hicks, will shortly be reunited with his father, Terry, whos on his way to Cuba to attend his son #39;s trial before a US military commission.

Рисунок 2.1. Приклади текстів з AG News Classification Dataset

2.3. Підготовка вхідних даних

Щоб робота з даними була більш ефективною потрібно розробити модуль, який виконуватиме попередню обробку текстових даних. Потрібно буде мати доступ до функції, яка отримує на вхід текст та виконує наступні дії:

- Токенізація тексту. Весь текст перетворюється до нижнього регістру та розбивається на окремі токени. Токени, які складаються лише з чисел або знаків пунктуації або не містять літер видаляються оскільки вони в більшості випадків не несуть суттєвого значення для подальшого аналізу.
- Видалення стоп-слів. Зі списку всіх токенів видаляються всі значення, які мають низьку значущість для визначення категорії тексту. Це допомагає зменшити кількість неважливої інформації та покращити якість аналізу.
- Лематизація або стемінг токенів. Допомагає значно зменшити розмір словника та покращити точність аналізу шляхом зведення різних форм одного слова до єдиного кореня.

Всі ці операції мають допомогти зменшити розмірність текстових даних, виокремити важливі ключові слова та поліпшити якість аналізу.

Поділ набору даних також є дуже вагомим етапом для ефективного навчання та коректної оцінки продуктивності моделі. У більшості випадків достатньо,

дотримуючись рівномірних пропорцій класів, розділити всі наявні екземпляри на три різні категорії:

- Навчальний дата сет, який використовується для безпосереднього тренування й підбору оптимальних ваг та коефіцієнтів моделі.
- Валідаційний дата сет, який використовується для перевірки моделі після кожного циклу тренування й може допомогти підібрати оптимальні гіперпараметри для моделі.
- Тестувальний дата сет, який необхідний для неупередженої оцінки кінцевих результатів моделі. [20]

Правильний розподіл даних дуже важливий, так як при неправильному розподілі модель може страждати від однієї з поширених проблем:

- Недостатнє навчання (underfitting). Трапляється коли модель мала недостатньо мало даних для навчання і відповідно демонструє низьку точність як і на тренувальних даних, так і на нових.
- Надмірне навчання (overfitting). Трапляється коли модель занадто сильно підлаштовується саме до тих даних, на яких вона проходила процес

тренування. В таких випадках модель демонструє високу точність на навчальних даних, але низьку точність на нових даних. [21]

У цьому випадку використаний дата сет розбивається на категорії таким чином:

- Більшу частину, а саме 96000 текстів (80% від загальної кількості), використано для безпосереднього тренування моделі.
- Для валідації виділено 10% від загальної кількості текстів, тобто 12000.
- Така само 10% текстів виділено для тестування фінальних результатів.

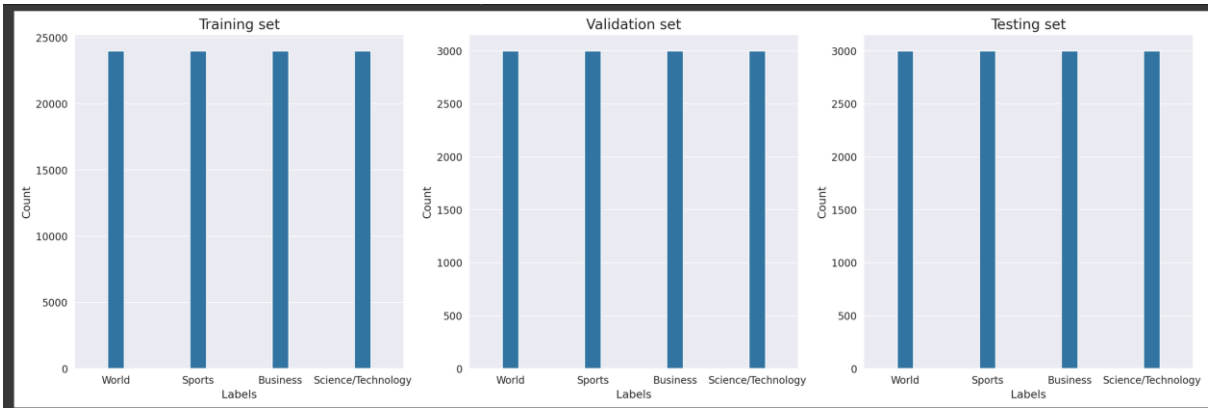

Рисунок 2.2. Розподіл даних

У деяких випадках також доречно було б перевірити наскільки ефективно моделі можуть навчатись, маючи досить обмежений обсяг даних для тренування. Для цього дата сет також розбивається на три категорії таким чином:

- 0.1% (120 екземплярів) для тренування.
- 0.1% (120 екземплярів) для валідації.
- 99.8% (117600 екземплярів) для тестування.

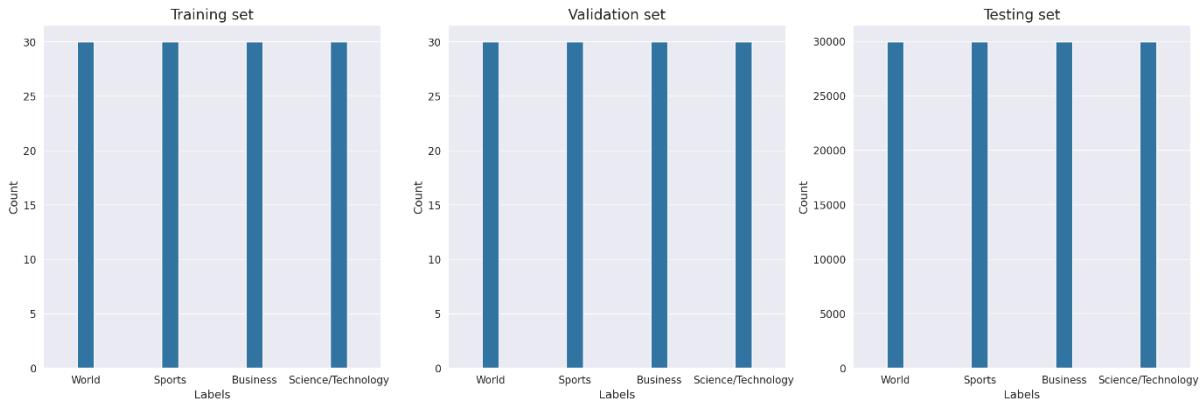

Рисунок 2.3. Розподіл даних з малою кількістю даних для тренування

2.4. Обрані моделі

Для класифікації буде використовуватись комбінація трьох моделей:

- Naive Bayes.
- SVM.
- RNN.

Naive Bayes є швидким алгоритмом, який потребує найменше ресурсів серед усіх трьох вищезгаданих. Якщо припущення про незалежність між ознаками справджується, то може демонструвати доволі ефективні результати, що робити його гарним вибором для таких задач як, наприклад фільтрація спаму. Бібліотека scikit-learn має клас MultinominalNB, який реалізує наївний баєсів класифікатор для багатономіальних моделей, який ми можемо використати в нашій роботі.

SVM також відносно економний в плані використання ресурсів комп’ютера, але не настільки як наївний баєсів класифікатор. Також використання даного алгоритму може демонструвати доволі ефективні результати навіть після тренування на відносно невеликому обсягу даних. Бібліотека scikit-learn також надає реалізацію опорно-векторної машини у вигляді класу LinearSVC.

RNN серед трьох обраних алгоритмів найбільш затратний в плані ресурсів, проте в загальному випадку після тренування на достатній кількості даних демонструє найкращі результати. Підходить для завдань де є важливим контекст,

таких як аналіз настроїв або подібних. Для того, щоб уникнути проблеми затухання градієнту буде використовуватись LSTM.

BERT та LLM використовуватись не будуть, оскільки вони потребують більш потужних обчислювальних ресурсів та великої кількості часу для тренування. Також не будуть використатись будь-які реалізації систем на основі правил, адже ми не зможемо адаптувати старі правил для нових категорій у випадку, якщо користувач натренує модель на власних текстах. Загалом згаданих трьох методів має бути достатньо, щоб переваги одного методу могли компенсувати недоліки іншого, не використовуючи при цьому надмірну кількість ресурсів комп'ютера.

Потрібно надати можливість використовувати будь-яку комбінацію моделей для тренування на користувацьких текстах. Наприклад для швидкого результату можна навчити тільки Naive Bayes, а для більш точних результатів натренувати всі три моделі. Так само й для класифікації повинна бути можливість використовувати будь-яку комбінацію натренованих моделей. Для цього будуть отримуватись відсоткові передбачення категорії з кожного класифікатора, а фінальний результат за замовчуванням буде середнім арифметичним усіх значень. Але у користувача повинна бути можливість зменшити вплив на результат класифікації, тих класифікаторів, які, на його думку, показують гірший результат, та навпаки – збільшити для тих, які показують кращий результат. Тому остаточний результат класифікації тексту буде визначатись наступною формулою:

$$x = a * w_c + b * w_b + c * w_c$$

- де x – результат системи класифікації
- a – результат класифікації з використанням Naive Bayes
- w_c – "вага" результату класифікації Naive Bayes, ціле число більше 0, що визначається користувачем
- b – результат класифікації з використанням SVM
- w_b – "вага" результату класифікації SVM
- c – результат класифікації з використанням RNN

- w_c – “вага” результату класифікації RNN

Таким чином, з використанням трьох коефіцієнтів, буде можливість коригувати систему класифікації для більш оптимальних результатів.

РОЗДІЛ 3 : РЕАЛІЗАЦІЯ СИСТЕМИ КЛАСИФІКАЦІЇ

3.1. Структура системи

Створено UML діаграму класів, яка схематично демонструє основні класи, які будуть реалізовані в застосунку, їхню взаємодію та функції, які вони зможуть виконувати.

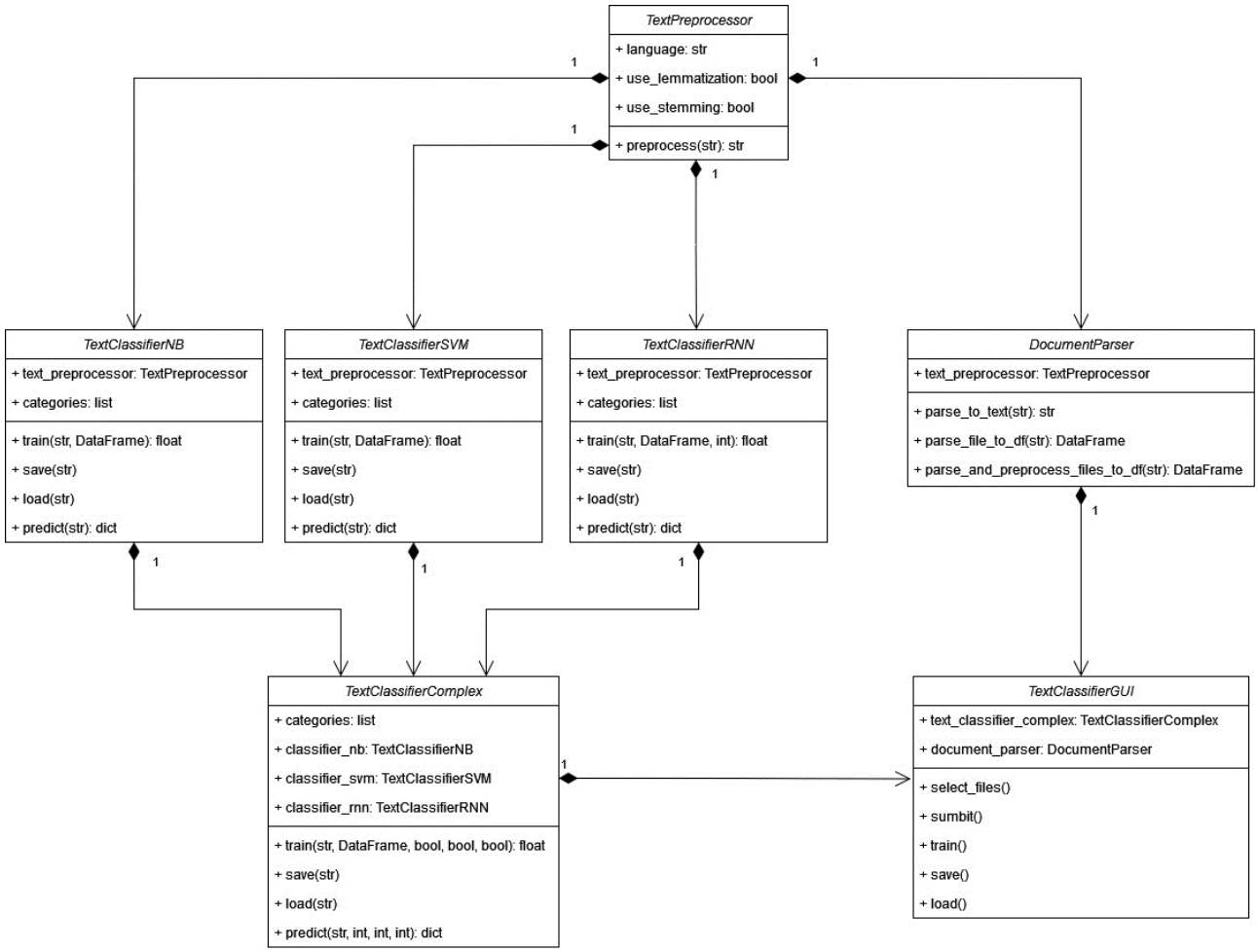

Рисунок 3.1. Діаграма класів

Клас TextPreprocessor дає можливість обрати мову та опції обробки (стемінг та лематизація), а також надає метод preprocess, який прийматиме на вхід не оброблений текст, а повертатиме вже оброблений.

TextClassifierNB – класифікатор, що використовує Naive Bayes. Має доступ до обробника тексту та містить список назв категорій, які здатен класифікувати. Метод train приймає на вхід або шлях до директорії з якої будуть діставатись й оброблюватись текстові файли для тренування або ж уже оброблений набір даних у вигляді DataFrame. Після виконання тренування повертається прогнозоване

значення точності. Методи save виконує збереження моделі в вказану директорію, а load – завантаження. Метод predict приймає на вхід не оброблений текст та повертає результат класифікації.

TextClassifierSVM – клас, що працює схожим чином з попереднім класом, але використовує SVM.

TextClassifierRNN – класифікатор, що використовує RNN модель. На відміну від інших класифікаторів, також при тренуванні можна зазначити кількість епох.

DocumentParser – клас, що спрощує роботу з документами. Містить набір методів, які витягуються текст з txt, docx чи pdf файлів у потрібному форматі.

TextClassifierComplex – має доступ до всіх трьох класифікаторів. При тренуванні є можливість обрати, які з моделей використовувати. Метод predict використовує коефіцієнти для кожного класифікатора при обчисленні результатів.

TextClassifierGUI – клас, який управляє графічним інтерфейсом. Працює з DocumentParser для отримання вмісту документів та з TextClassifierComplex для його класифікації.

Для кращого розуміння процесу взаємодії графічного інтерфейсу з іншими сутностями варто створити UML діаграму послідовності, яка відображатиме простий випадок використання системи для класифікації файлів користувача.

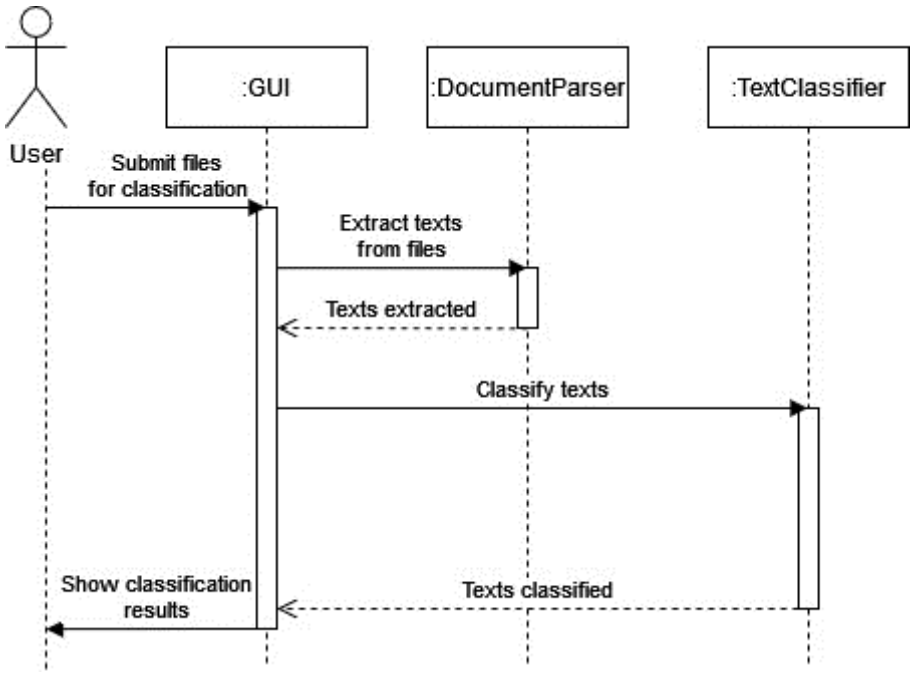

Рисунок 3.2. Діаграма послідовності для процесу класифікації

Користувач обирає файли через GUI та запускає процес класифікації. DocumentParser витягує з файлів текстову інформацію, після чого вона передається в класифікатор тексту для визначення категорії. Результати класифікації

повертаються й демонструюся користувачу у форматі відповідному обраним в графічному інтерфейсі опціям.

Так само демонструється процес тренування моделі на файлах користувача та її збереження.

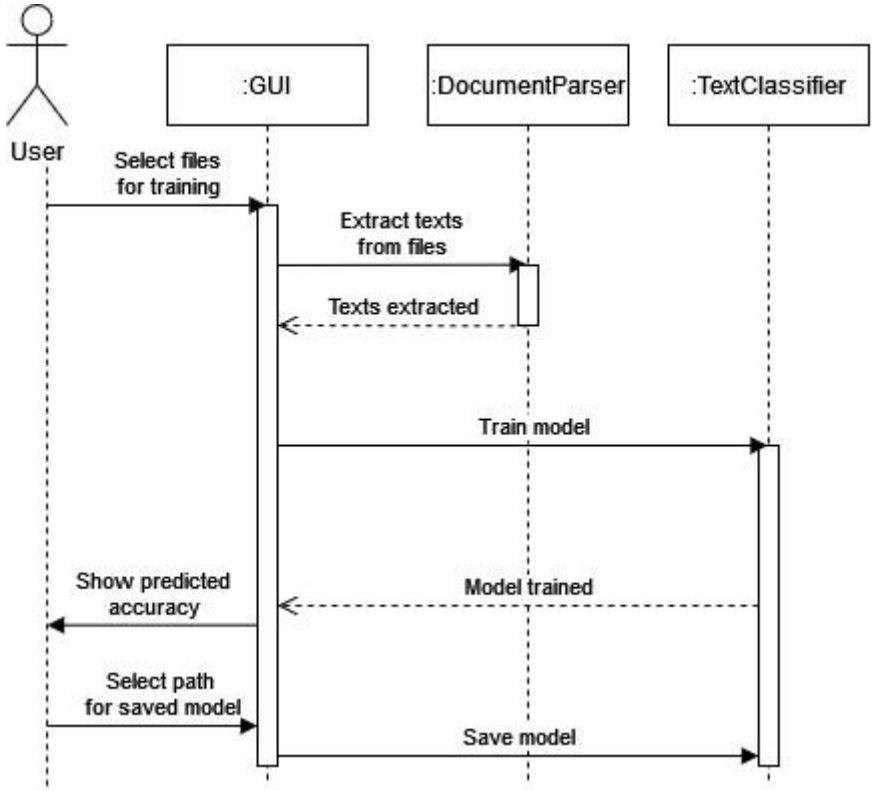

Рисунок 3.3. Діаграма послідовності для процесу тренування моделі

Користувач обирає файли для тренування (файли мають бути відсортовані по піддиректоріях відповідно до назв класів чи категорій). Текст з файлів витягується та передається до класифікатора для тренування. Після тренування користувач зможе побачити значення прогнозованої точності класифікації. Є можливість зберегти натреновану модель в обрану директорію для того, щоб була можливість завантажити її пізніше.

3.2. Обробка тексту

Клас TextPreprocessor розроблено з використанням базових функцій бібліотеки NLTK. При ініціалізації є можливість обрати мову (доступна англійська та експериментальна реалізація для української), а також можна увімкнути опції стемінгу чи лематизації.

```
class TextPreprocessor:
    Andrii Dubovyk *
    def __init__(self, language='english', use_lemmatization=False, use_stemming=True):
        self.language = language
        self.use_stemming = use_stemming
        self.use_lemmatization = use_lemmatization
        if language == 'ukrainian':
            self.__setup_ukrainian_language()
        else:
            self.__setup_default_language()
```

Рисунок 3.4. Ініціалізація класу TextPreprocssor

Для англійської мови застосовується PorterStemmer, WorldNetLemmatizer та список стоп-слів, який надає бібліотека NLTK. Для обробки ж текстів української мови використовується відповідний список стоп-слів із текстового файлу. Як стемер використано бібліотеку UkStemmer [22], а для лематизації використовується pymorhy2 [23].

```
def __setup_ukrainian_language(self):
    stopwords_ua = pandas.read_csv( filepath_or_buffer: "resources/stopwords_ua.txt", header=None, names=['stopwords'])
    self.stop_words = list(stopwords_ua.stopwords)
    if self.use_stemming:
        self.stemmer = uk_stemmer.UkStemmer()
        self.stem = self.stemmer.stem_word
    if self.use_lemmatization:
        self.lemmatizer = pymorphy2.MorphAnalyzer(lang='uk')
        self.lemmatize = lambda word: self.lemmatizer.parse(word)[0].normal_form
```

Рисунок 3.5. Фрагмент коду, що демонструє налаштування для української мови

Також для того, щоб полегшити роботу з файлами створено модуль DocumentParser, який може одразу діставати текстову інформацію з файлів формату txt, pdf та docx, які відсортовано по директоріях відповідно до її класів. Текст оброблюється й повертається в зручному форматі (яким в цьому випадку є DataFrame з бібліотеки pandas).

```python
@staticmethod
def parse_files_to_df(directory):
    classes = []
    for root, dirs, files in os.walk(directory):
        for d in dirs:
            classes.append(d)
    data = []
    for category in classes:
        category_dir = os.path.join(directory, category)
        for filename in os.listdir(category_dir):
            file_path = os.path.join(category_dir, filename)
            text = DocumentParser.parse_to_text(file_path)
            data.append({'label': category, 'text': text})
    return pd.DataFrame(data)

1 usage  Andrii Dubovyk
@staticmethod
def parse_and_preprocess_files_to_df(directory):
    df = DocumentParser.parse_files_to_df(directory)
    text_preprocessor = TextPreprocessor(language='english', use_stemming=False, use_lemmatization=True)
    df['text'] = df['text'].apply(lambda txt: text_preprocessor.preprocess(txt))
    return df
```

Рисунок 3.6. Фрагмент коду, що демонструє процес завантаження та обробки текстових файлів

3.3. Реалізація алгоритмів класифікації

Для реалізацію класифікатора Naive Bayes використано клас MutlinominalNB з бібліотеки scikit-learn.

```python
# Define model
self.model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('nb', MultinomialNB()),
])

# Train model
self.model.fit(x_train['text'], x_train['label'])
```

Рисунок 3.7. Визначення моделі Naive Bayes та її тренування

Схожим чином для SVM використовується функція LinearSVC з бібліотеки scikit-learn. Але звичайна реалізація LinearSVC всього лиш може видавати результат у вигляді однієї найбільш вірогідної категорії. Щоб отримувати відсоткове значення для кожної з категорій, ми робимо обгортку за допомогою CallibratedClassifierCV.

```
# Define model
self.model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('svc', CalibratedClassifierCV(LinearSVC())),
])

# Train model
self.model.fit(x_train['text'], x_train['label'])
```

Рисунок 3.8. Визначення моделі SVM та її тренування

Методи класифікації Naive Bayes та SVM доволі просто реалізовуються за допомогою відповідних функцій бібліотеки scikit-learn, але реалізація RNN за допомогою TensorFlow надає нам більше можливостей для модифікації та налаштувань задля покращення ефективності моделі.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVec  (None, None)            0
 torization)

 embedding (Embedding)       (None, None, 64)          640000

 bidirectional (Bidirectiona  (None, None, 128)        66048
 l)

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dense (Dense)               (None, 64)                8256

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 714,820
Trainable params: 714,692
Non-trainable params: 128
```

Рисунок 3.9. Архітектура RNN моделі з використанням LSTM

Можемо детально розглянути архітектуру розробленої RNN (LSTM) моделі. Всього вона складається 8 шарів:

1. TextVectorization. Перетворює оброблений текст на послідовність індексів токенів. Використовує словник розміром 10000, що є більш ніж достатньо з врахуванням того, що текст проходить обробку, яка включає лематизацію.
2. Embedding. Шар вбудування, який зберігає один вектор на слово. При виклику він перетворює послідовності індексів слів на послідовності векторів. Ці вектори можна тренувати. Після навчання на достатній кількості даних слова зі схожими лексичними значеннями часто мають подібні вектори.

Має параметр mask_zero=True, що дозволяє ефективно працювати з текстами різних розмірів.

3. Bidirectional. Є обгорткою для RNN, що обробляє послідовний вхід у двох напрямках. В цьому випадку використовує LSTM шар з 64 нейронами для збереження контексту.
4. GlovalMaxPooling1D. Рівень об’єднання, який зменшує чутливість до особливостей, таким чином створюючи більш узагальнені дані для кращих результатів тренування.
5. Dense. Повнозв’язний шар, який використовує функцію активації relu та містить 64 нейрони.
6. BatchNormalization. Використовується для підвищення швидкості та стабільності навчання штучних нейронних мереж. Це досягається шляхом нормалізації вхідних даних шарів через повторне центрування та масштабування.
7. Dropout. Відкидає випадковим чином деякі нейрони, може допомогти частково уникнути надмірного тренування, яке відбувається, коли модель надто добре або занадто довго тренувалась на заданому наборі даних і її не вдається демонструвати хороші результати при застосуванні до нових даних. В цьому випадку використовується значення rate 0.2, що означає що 20% випадкових нейронів будуть ігноруватись. Також варто зауважити, що цей шар використовується лише при тренуванні, тому під час справжніх передбачень моделі нейрони не будуть ігноруватись.
8. Dense. Повнозв’язний шар, який містить 4 нейрони, що відповідає кількості категорій в нашому дата сеті. Використовує функцію активації softmax та повертає розподіл ймовірностей визначених категорій.

Дана модель компілюється з наступними параметрами:

- sparse_categorical_crossentropy використовується для обчислення втрат між прогнозованими та справжніми мітками. Це доцільний вибір для задачі класифікації з кількома класами, де мітки не кодуються як one-hot

вектори, але представлені цілими числами. Дозволяє отримати список найбільш ймовірних категорій.

- Використовується оптимізатор Adam зі швидкістю навчання 1e-4. Оптимізатор Adam ефективно пристосовує швидкість навчання для кожного параметра зокрема на основі оцінок першого та другого моментів градієнта.
- Метрика "accuracy" використовується для оцінки точності моделі під час навчання. Ця метрика вимірює відсоток правильно класифікованих зразків у всій кількості екземплярів.

Також використовується функція зворотного виклику EarlyStopping, яка зупиняє тренування коли точність класифікації моделі для валідаційного дата сету перестає зростати. В цьому випадку ця функція використовується з наступним параметрами:

- Значення, за яким потрібно робити нагляд: val_accuracy (точність моделі на валідаційному дата сеті).
- Кількість епох без покращення, після яких навчання буде припинено: 2.
- Мінімальна зміна значення, що контролюється, яка буде вважатися покращенням (тобто абсолютна зміна менше min_delta, вважатиметься відсутністю покращення): 0.001.
- Коли ця функція спрацьовуватиме, ми будемо бачити про це повідомлення в консолі.
- Після спрацювання функції буде відновлено той стан моделі, в якому вона мала найкраще значення точності.

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
                                                  patience=2,
                                                  min_delta=1e-3,
                                                  verbose=1,
                                                  restore_best_weights=True)
```

Рисунок 3.10. Функція зворотного виклику EarlyStopping

Ця функція в багатьох випадках відбирає в нас потребу підбирати оптимальну (для уникнення недостатнього або надмірного тренування) кількість епох вручну, дозволяючи просто запустити тренування на великій кількості епох і дочекатись доки точність перестане зростати.

В підсумку після тренування можемо побачити графіки зміни точності та втрат (синім кольором показані результати для тренувальних даних, оранжевим – для валідаційних). Тренування відбувалось впродовж п’яти епох, доки точність не перестала зростати, після чого було відновлено стан моделі, в якому вона демонструвала найкращу точність на валідаційних даних (в цьому випадку це третя епоха).

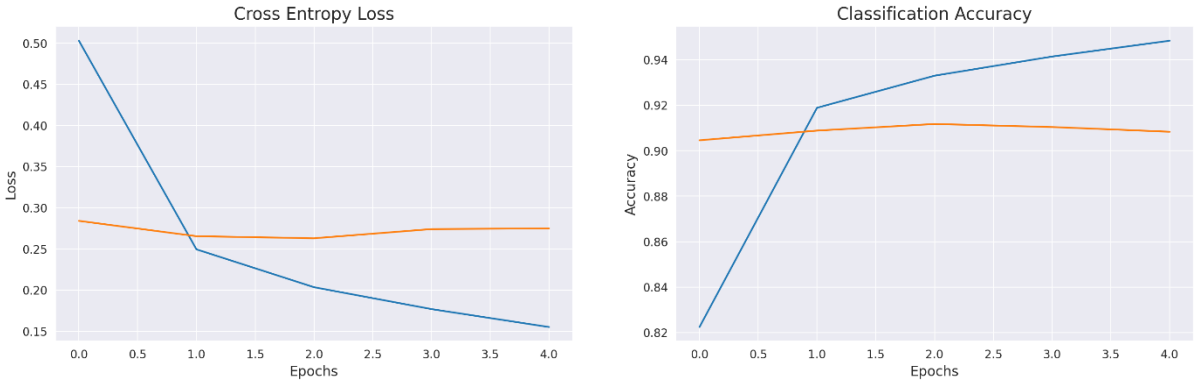

Рисунок 3.11. Втрати та точність LSTM моделі впродовж тренування

Також були протестовані деякі інші часто використовувані оптимізатори [0], такі як Adagard, Adadelta і RMSprop, з метою визначити їх вплив на ефективність натренованої моделі. З іншими оптимізаторами, використовуючи стандартну швидкість навчання зі значенням 0.001, після тренування максимально вдалось досягти таких значень:

- Adagard – 90.22% точності та значення втрат 0.2933.

- Adadelta – 90.71% точності та значення втрат 0.2764.

- RMSprop – 91.93 % точності та значення втрат 0.2524.

Тільки з оптимізатором RMSprop вийшло досягнути дещо кращого результату ніж з Adam, в той час, як оптимізатори Adagard та Adadelta не змогли показати значення

точності вище 91%, тому було прийнято рішення далі використовувати саме цей оптимізатор. Також тестувалось тренування моделі з іншою (0.01 та 0.0001) швидкістю навчання, відмінною від стандартної 0.001, але якихось суттєвих змін в результатах помічено не було, тому швидкість навчання зі стандартним значенням 0.001 була залишена як оптимальна.

3.4. Система класифікації з використання комбінації алгоритмів

Кожну реалізацію алгоритму класифікації з попереднього підрозділу було додано до окремих відповідних класів: TextClassifierNB, TextClassifierSVM, TextClassifierRNN. Кожен з класів має наступні методи необхідні для роботи:

- train – приймає на вхід або директорію з текстами або DataFrame з обробленими текстами, для RNN моделі може також приймати кількість епох для навчання, тренує модель й повертає значення точності класифікації у відсотках.
- save – приймає на вхід директорію, до якої зберігає навчену модель (NB та SVM моделі зберігаються у форматі pkl, RNN – у форматі keras, окремо в json файл зберігаються назви категорій).
- load – приймає на вхід директорій з якої завантажує попередньо збережену модель.
- predict – приймає на вхід текст користувача для якого визначає категорії, результат повертає в форматі відсортованого Python словника, де ключі це назви категорій, а значення – ймовірність належності до категорії від нуля до одиниці.

Аналогічно розроблено клас TextClassifierComplex, що використовує всі вищезгадані класи та містить такі методи:

- train – схожий на методи окремих класів, за допомогою булевих параметрів use_nb, use_svm, use_rnn є можливість визначати, які моделі будуть тренуватись.
- save – зберігає всі натреновані моделі в обрану директорію.
- load – завантажує всі моделі, які збережені в обраній директорії.
- predict – аналогічний до методу окремих класів, має параметри nb_weight, svm_weight, rnn_weight значення, яких коригує вплив передбачення відповідної моделі на загальний результат.

3.5. Графічний інтерфейс

Був розроблений графічний інтерфейс, який не тільки дозволяє користувачу ефективно використовувати система з натренованими моделями, які навчені на AG

News Classification Dataset, а й натренувати систему на власних файлах, зберегти результати навчання та використовувати її надалі. Одразу при запуску застосунку завантажуються вже заздалегідь навчені моделі.

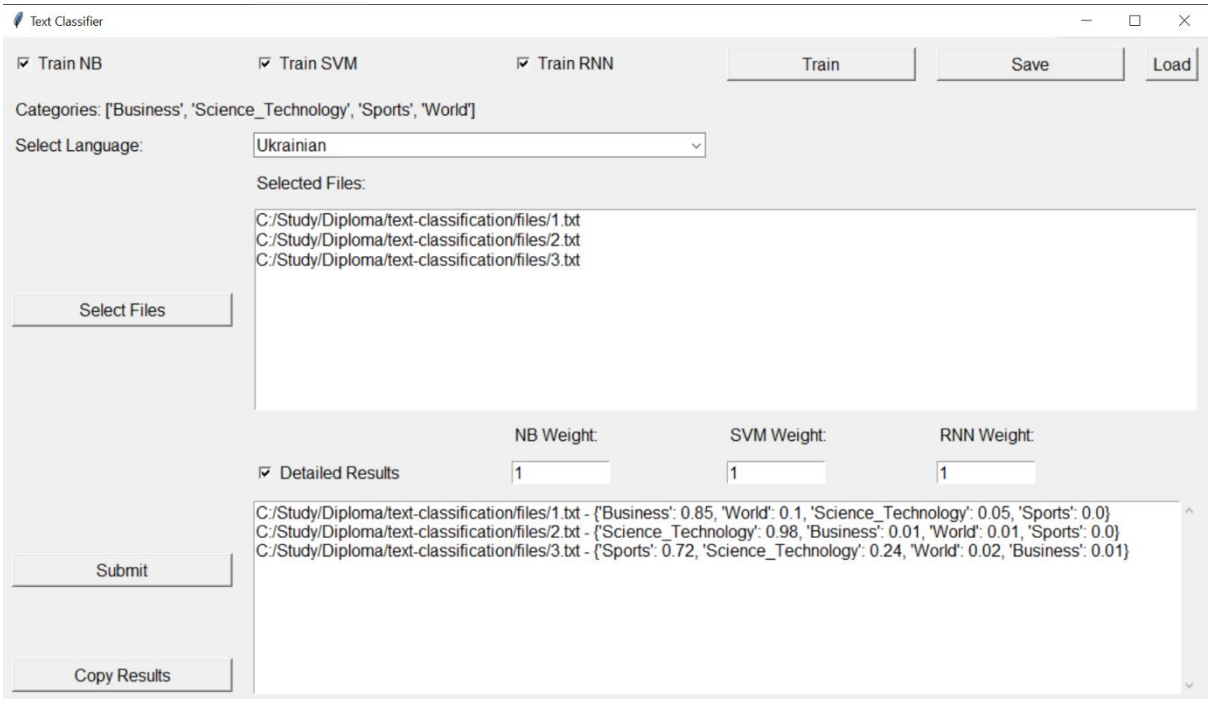

Рисунок 3.12. Графічний інтерфейс

Вгорі вікна програми містяться елементи, які стосуються тренування, збереження та завантаження моделі користувача. Прапорці “Train NB”, “Train SVM”, “Train RNN” відповідають за те, які моделі будуть тренуватись. Кнопка “Train” відкриває діалогове вікно, що дозволяє обрати директорію з текстовими документами (txt, pdf або docx) для тренування, потім запускається сам процес тренування обраних моделей. Після тренування в діалоговому вікні буде показано приблизне значення точності класифікації (середнє арифметично всіх використаних моделей на тренувальному наборі даних). Варто зазначити, що текстові документи в обраній директорії мають бути розміщені належним чином – кожен документ має знаходиться в піддиректорії назва якої відповідає визначені категорії тексту. Кнопка “Save” дозволяє зберегти модель в одну з директорій комп’ютера, а кнопка “Load” – завантажити попередньо збережену модель. Текст

“Categories: [перелік категорій]” демонструє, які категорії здатна розпізнавати модель, яка зараз використовується.

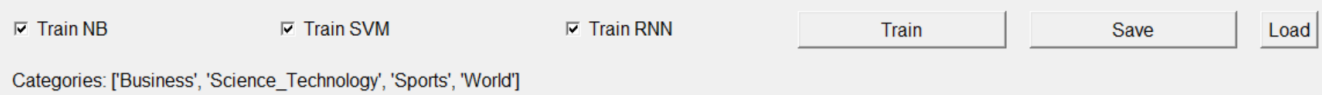

Рисунок 3.13. Елементи графічного інтерфейсу, які стосуються управління моделлю користувача

Далі розміщена група елементів графічного інтерфейсу, що стосуються завантаження документів для класифікації. Є можливість обрати мову наданих текстів, що необхідно для правильної класифікації. За замовчуванням використовується англійська мова, але є можливість обрати українську, в такому разі всі обрані тексти користувача будуть автоматично перекладатись через Google Translate на англійську мову перед тим, як модель буде робити передбачення їх категорії. Це дозволяє не тренувати модель заново для іншої мови, а використовувати одну і ту ж модель для розпізнавання текстів мов відмінних від англійської. Документи для класифікації обираються в діалоговому вікні, що з’являється після натискання “Select Files”. Повний шлях до обраних файлів демонструється в відповідному текстовому полі.

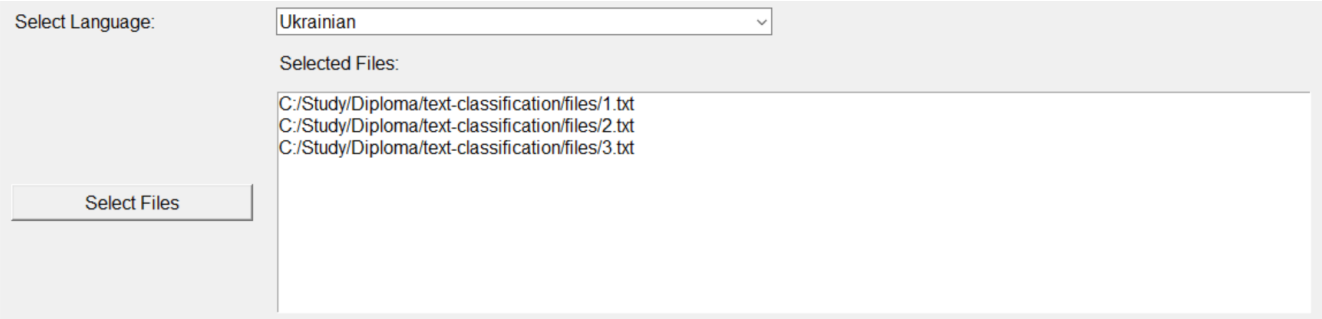

Рисунок 3.14. Елементи графічного інтерфейсу, які стосуються завантаження документів

Остання група елементів інтерфейсу відповідає за саму класифікацію, та параметри системи, що при цьому використовуються. Поля “NB Weight”, “SVM Weight”, “RNN Weight” дозволяють коригувати вплив кожної окремої моделі на загальний результат. Прапорець “Detailed Results” впливає на формат відображення результатів”, якщо він відмічений, то для кожного документа буде вказано відсоткове значення належності до категорій, якщо ні – лише одна найбільш

вірогідна категорія (без значення відсотків). Кнопка “Submit” запускає процес визначення категорії, а “Copy Results” дозволяє швидко та зручно скопіювати результати в буфер обміну.

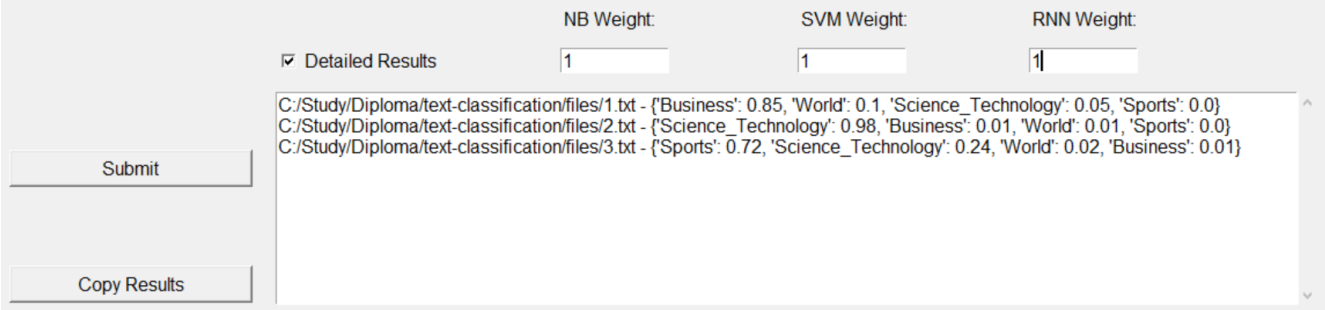

Рисунок 3.15. Елементи графічного інтерфейсу, які стосуються безпосередньої класифікації

3.6. Оцінка результатів

3.6.1. Результати категоризації новин

Найкращі результати для використаного набору даних демонструвала саме RNN модель, тому в цьому випадку детальніше розглянемо саме її результати. На тестових даних розроблена модель у остаточному її варіанті показала наступні результати:

- Точність: 91.92500114440918 %
- Втрати: 0.26782718300819397

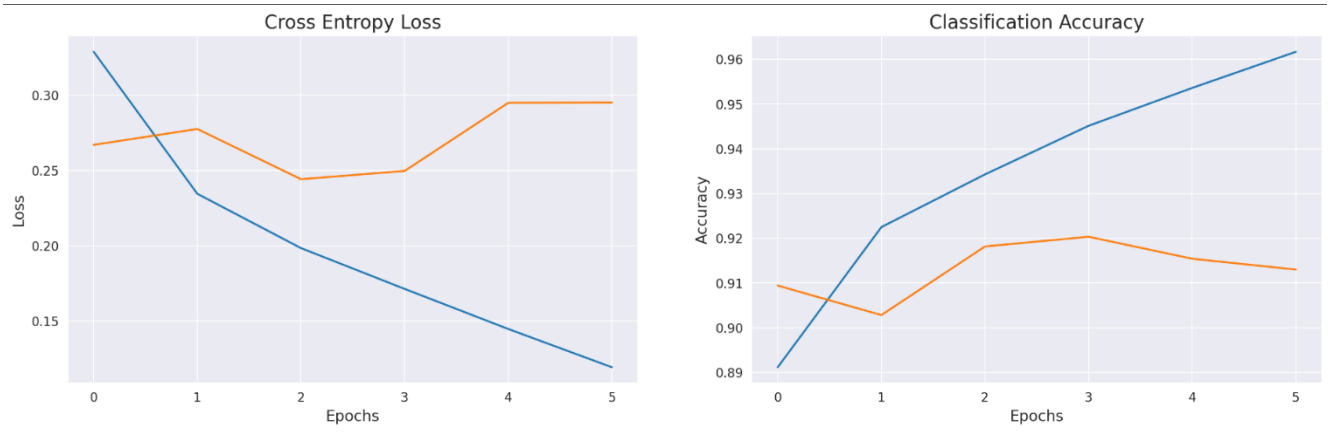

Рисунок 3.16. Графіки втрат та точності RNN моделі в її остаточному вигляді

Також у звіті про класифікацію можна побачити деякі показники для кожної з чотирьох категорій окремо:

- Precision - кількість справжніх позитивних результатів, поділена на кількість прогнозованих позитивних результатів.
- Recall - кількість справжніх позитивних результатів, поділена на загальну кількість фактичних позитивних результатів.
- F1-score - середнє гармонічне між precision та recall.

	precision	recall	f1-score	support
World	0.89	0.89	0.89	3000
Sports	0.90	0.89	0.89	3000
Business	0.96	0.98	0.97	3000
Science/Technology	0.93	0.91	0.92	3000
accuracy			0.92	12000
macro avg	0.92	0.92	0.92	12000
weighted avg	0.92	0.92	0.92	12000

Рисунок 3.17. Звіт про класифікацію

Матриця невідповідностей демонструє скільки екземплярів однієї категорії було визначено іншою, невідповідною категорією під час передбачень категорій для тестового набору даних. Можемо побачити, що найчастіше модель плутає між собою категорії “World” та “Sports”. Також відносно часто плутає “Science/Technology” з “World” та “Sports” з “Science/Techonlogy”. Категорія ж “Business” визначається найбільш впевнено і плутається з усіма іншими доволі рідко.

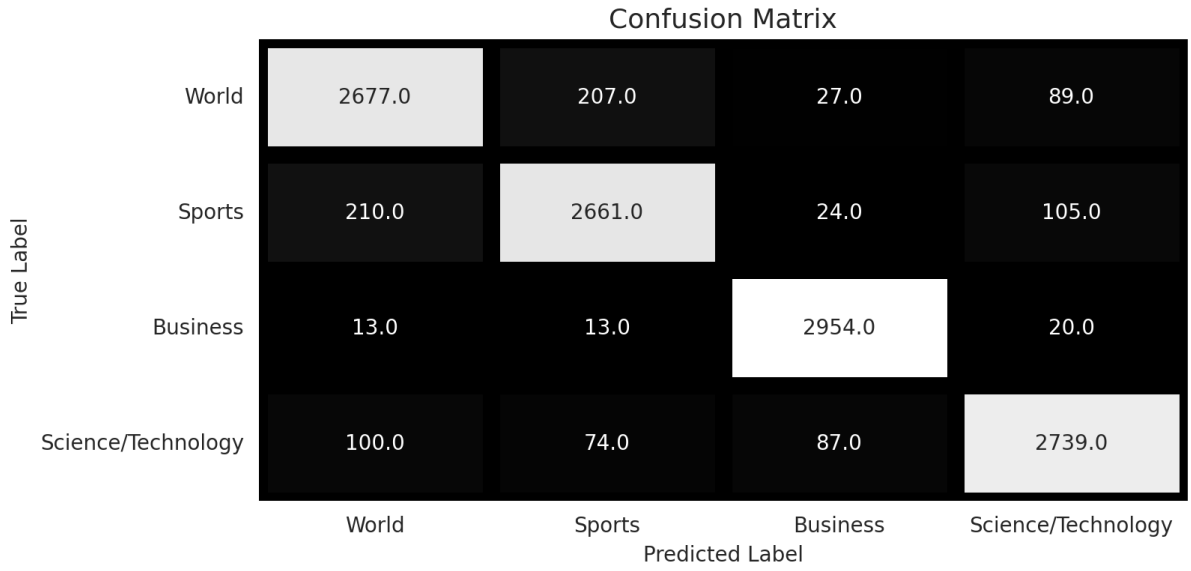

Рисунок 3.18. Матриця невідповідностей

Також демонструється приклад передбачення моделі на тексті, який не входить у використаний датасет і дещо відрізняється стилістично. На скриншоті показано програмний код даного процесу, в консолі виведено результати: оброблений текст, а також вірогідність кожної категорії починаючи з найвищої. Бачимо, що на даному прикладі модель з точністю близько 96% класифікує тематику цього тексту як “Science/Technology”.

```
sample_text = """
In the realm of technology, artificial intelligence (AI) stands as a revolutionary force, reshaping industries, augmenting human capabilities, and fueling innovat
Through advanced algorithms and computational power, AI systems can analyze vast datasets, recognize patterns, and make autonomous decisions with unprecedented ac
From virtual assistants streamlining daily tasks to self-driving cars navigating complex environments, the potential applications of AI are vast and continually e
While unlocking new possibilities, AI also raises ethical and societal considerations, prompting discussions on privacy, bias, and the future of work in an increa
"""

# Preprocess the text sample (replace this with your actual preprocessing steps)
processed_text =  text_preprocessor.preprocess(sample_text)
print(processed_text)

# Make predictions
predictions = model.predict(np.array([processed_text]))

predicted_probabilities = predictions[0]
# Create a list of tuples containing category index and probability
category_prob_tuples = [(i, prob) for i, prob in enumerate(predicted_probabilities)]
# Sort the list of tuples by probability in descending order
sorted_category_prob_tuples = sorted(category_prob_tuples, key=lambda x: x[1], reverse=True)
# Display the results
for i, (category_index, probability) in enumerate(sorted_category_prob_tuples):
    print(f"{i+1}. Category: {categories[category_index]}, Probability: {probability}")
```

```
realm technology artificial intelligence ai stand revolutionary force reshaping industry augmenting human capability fueling innovation advanced algorithm comput
1/1 [==============================] - 3s 3s/step
1. Category: Science/Technology, Probability: 0.9636746048927307
2. Category: Business, Probability: 0.0267433300614357
3. Category: World, Probability: 0.00951559841632843
4. Category: Sports, Probability: 6.639408093178645e-05
```

Рисунок 3.19. Передбачення моделі

Для порівняння ефективності навченої моделі категорії тексту для 200 екземплярів з цього ж набору даних (AG News Classification Dataset) було визначено

через ChatGPT версії 3.5. В результаті правильно розпізнано було лише 176 категорій, що показує точність зі значенням близько 88%. Реалізована ж в даній роботі RNN (LSTM) модель демонструє приблизно на 4% кращий результат (майже 92%). Таким чином можна зробити припущення, що спеціалізовані моделі, крім того, що є значно ефективнішими в плані використання ресурсів, також можуть демонструвати кращий результат в класифікації текстів, ніж LLM.

З другим поділом дата сету, який містить обмежений обсяг даних для тренування, моделі показали такий результат:

- Модель з Naive Bayes сягнула значення точності в 68.34%.
- SVM модель досягла 70.25% точності.
- LSTM модель продемонструвала точність передбачення всього лише 24.98%.

Це демонструє перевагу моделей SVM та Naive Bayes в ситуації з недостатньою кількістю даних для тренування.

3.6.2. Результати категоризації новин (альтернативний дата сет)

Також було перевірено наскільки ефективно система може навчатись на інших текстах. З kaggle.com в цих цілях було звантажено набір даних з назвою “(10)Dataset Text Document Classification”. Цей дата сет містить по сотні текстів кожної з десяти категорій: “бізнес”, “розваги”, “їжа”, “графіка”, “історія”, “медицина”, “політика”, “космос”, “спорт” та “технології”. На завантаження та обробку текстів з даних файлів знадобилось 4.78 секунди. Моделі справились даною задачею наступним чином:

- Naive Bayes – демонструє 97 % точності витративши 0.2 секунди на навчання.
- SVM – демонструє 98 % точності витративши 0.59 секунди на навчання.
- RNN – демонструє 91 % точності витративши 974 секунди на навчання.
- Комбінація всіх моделей (з однаковими вагами) має точність 94.5 %.

3.6.3. Результати фільтрації спаму

Щоб перевірити, як система справляється з такою задачею як фільтрація спаму, було використано Spam Mails Dataset з kaggle.com, який містить 5171 зразків тексту з визначеними спам повідомленнями. Таким чином моделі справляються з даною задачею з такою точністю та витраченим часом на навчання (без врахування 9.53 секунд на завантаження файлів та обробку тексту):

- Naive Bayes – 92.07 % та 0.41 секунди.
- SVM – 99.23 % та 0.54 секунди
- RNN – 98.07 % та 1153 секунди.
- Комбінація всіх моделей (з однаковими вагами) має точність 99.13 %

Отже, система може бути швидко адаптована й до такої задачі. Майже за 10 секунд можна отримати SVM модель, яка зможе розпізнавати типові спам-повідомлення з точністю близько 99 %.

3.6.4. Результати аналізу настроїв

Так само були протестовані можливості системи при розпізнаванні емоційної забарвленості тексту. Для цього використано Emotions dataset for NLP з kaggle.com, який містить 16 тисяч зразків тексту з визначеними емоціями (сум, злість, страх, радість, здивування, любов). Текст з файлів завантажується та оброблюється за 2.8 секунди. Моделі демонструють таку точність та час навчання:

- Naive Bayes – 56.99 % та 0.081 секунди.
- SVM – 83.09 % та 0.83 секунди
- RNN – 85.56 % та 13.35 секунд.
- Комбінація всіх моделей (з однаковими вагами) має точність 59.31 %. Але якщо проекспериментувати з коефіцієнтами, то можна отримати кращі результати, ніж при використанні кожної моделі окремо. Наприклад вдалось досягти точності 87.75 % використавши такі значення: 1 для Naive Bayes, 5 для SVM та 13 для RNN

В результаті навіть, якщо один або два алгоритми не можуть показати хороший результат для певної задачі, то майже завжди як мінімум одна модель може

ефективно класифікувати тексти з точністю понад 85 %. Таким чином система може застосуватись майже для всіх загальних випадків класифікації.

ВИСНОВКИ

У першому розділі було проведено аналіз предметної області. Розглянуто основні теоретичні поняття, визначено основні переваги та недоліки використання наявних методів класифікації текстів.

Другий розділ містить інформацію про вибір інструментів для вирішення проблеми та алгоритму розв’язку. Були обрані моделі, які будуть використовуватись в розробленій системі класифікації текстів, та описано яким чином їхні результаті будуть комбінуватись.

Третій розділ було присвячено практичній частині роботи, а саме розробці обробника тексту, реалізації різних алгоритмів класифікації текстів та більше детальній оцінці їхньої ефективності в різних умовах. В результаті було розроблено систему, що в використовує три алгоритми класифікації:

1. Naive Bayes.
2. Support Vector Machine.
3. LSTM.

Крім того, було розроблено графічний застосунок, що використовує натреновану на AG News Classification Dataset систему і здатен розпізнавати тексти (англійською або українською мовою) чотирьох категорій (“World”, “Sports”, “Science/Technology”, “Business”) з точністю близько 92 %. Також даний графічний застосунок дозволяє користувачу натренувати модель на власних текстах та категоріях, зберегти її та використовувати надалі.

Продовження досліджень та експериментів в даній сфері можуть призвести до підвищення ефективності створених моделей у розпізнаванні використаних в даній роботі або будь-яких інших категорій текстів. Також подальший розвиток може бути направлений на покращення зручності та функціональності розробленого графічного застосунку.

Список використаних джерел

1. Machine Learning Glossary [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/machine-learning/glossary
2. A Generative Learning Model for Heterogeneous Text Classification Based on Collaborative Partial Classifications / Zie Eya Ekolle and Ryuji Kohno – Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/13/14/8211
3. A Survey on Text Classification Algorithms: From Text to Predictions / Gasparetto, A.; Marcuzzo, M.; Zangari, A.; Albarelli – Режим доступу до ресурсу: https://www.mdpi.com/2078-2489/13/2/83
4. Understanding Text Classification in Python [Інтернет стаття] – Режим доступу до ресурсу: https://www.datacamp.com/tutorial/text-classification-python
5. Scikit-learn: Naive Bayes [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/naive_bayes.html
6. Support vector machine [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Support_vector_machine
7. Google voice search: faster and more accurate [Інтернет стаття] – Режим доступу до ресурсу: https://research.google/blog/google-voice-search-faster-and-more-accurate
8. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing [Інтернет стаття] – Режим доступу до ресурсу: https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processing
9. Large language model (LLM) [Інтернет стаття] – Режим доступу до ресурсу: https://www.growthloop.com/university/article/llm
10. ChatGPT a year on: 3 ways the AI chatbot has completely changed the world in 12 months [Інтернет стаття] – Режим доступу до ресурсу: https://www.euronews.com/next/2023/11/30/chatgpt-a-year-on-3-ways-the-ai-chatbot-has-completely-changed-the-world-in-12-months
11. A Hybrid Text Classification Approach for Analysis of Student Essays / C.P. Ros´e, A.Roque, D. Bhembe, K. Vanlehn – Режим доступу до ресурсу:

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7f7e133f636308b5f67600ad321335f716a7a14a

12. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org
13. Scikit-learn [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable
14. Natural Language Toolkit [Електронний ресурс] – Режим доступу до ресурсу: https://www.nltk.org
15. NumPy [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org
16. Pandas [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org
17. Matplotlib [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org
18. Seaborn [Електронний ресурс] – Режим доступу до ресурсу: https://seaborn.pydata.org
19. Kaggle [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com
20. A Guide to Data Splitting in Machine Learning [Інтернет стаття] – Режим доступу до ресурсу: https://medium.com/@datasciencewizards/a-guide-to-data-splitting-in-machine-learning-49a959c95fa1
21. ML | Underfitting and Overfitting [Інтернет стаття] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning
22. UkStemmer [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Desklop/Uk_Stemmer
23. Pymorphy2 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/pymorphy2/pymorphy2
24. Comparing automated text classification methods. International Journal of Research in Marketing(2019), Volume 36, Pages 20-38 / Jochen Hartmann, Juliana Huppertz, Christina Schamp, Mark Heitmann – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0167811618300545
25. Stemming vs Lemmatization in NLP [Інтернет стаття] – Режим доступу до ресурсу: https://nirajbhoi.medium.com/stemming-vs-lemmatization-in-nlp-efc280d4e845
26. Text Categorization Based on a Similarity Approach / Yang, Cha & Wen, Jun. (2007) – Режим доступу до ресурсу:

https://www.researchgate.net/publication/266577030_Text_Categorization_Based_on_a_Similarity_Approach

27. Toward any-language zero-shot topic classification of textual documents. Artificial Intelligence (2019), Volume 274, 133-150 / Yangqiu Song, Shyam Upadhyay, Haoruo Peng, Stephen Mayhew, Dan Roth – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0004370219300414
28. Unsupervised Text Classification with Topic Models and Good Old Human Reasoning [Інтернет стаття] – Режим доступу до ресурсу: https://medium.com/@power.up1163/unsupervised-text-classification-with-topic-models-and-good-old-human-reasoning-da297bed7362