Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів виявлення синтетичних текстів

(тема)

Виконав:
студент 2 курсу, групи СШІм-21-1
Безродний В.В.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Турута О.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Безродному Владиславу В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів виявлення синтетичних текстів

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, статті Інтернет-джерел, документація Keras та numpy, тексти, згенеровані штучним інтелектом

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної області

2) Огляд існуючих методів класифікації

3) Дослідження алгоритмів машинного навчання та штучних нейронних мереж

4) Експериментальне дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	04.04.2023	Виконано
2	Постановка задачі	08.04.2023	Виконано
3	Аналіз методів та підходів до класифікації текстів	15.04.2023	Виконано
4	Експериментальне оцінювання моделей	20.04.2023	Виконано
5	Написання пояснювальної записки	25.04.2023	Виконано
6	Попередній захист	15.05.2023	Виконано
7	Захист перед ЕК	18.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 53 с., 13 рис., 1 табл., 1форм., 1 дод, 27 джерел.

КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, ОБРОБКА МОВИ, СИНТЕТИЧНЫ ТЕКСТИ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ.

Об’єкт дослідження – методи машинного навчання для виявлення синтетичних текстів та текстів написаних людиною.

Предмет дослідження – методи класифікації за допомогою машинного навчання.

Мета дослідження – виявити найефективніші методи розпізнавання синтетичних текстів.

У кваліфікаційній роботі дослідження зосереджене на методах машинного навчання для виявлення синтетичних текстів та текстів, написаних людиною, з метою знайти найефективніші методи розпізнавання синтетичних текстів. Оскільки синтетичні тексти можуть бути важким завданням для класифікації, використання алгоритмів машинного навчання та штучного інтелекту може допомогти вирішити цю проблему.

Дослідження може включати порівняння різних методів та алгоритмів класифікації, оцінку їх точності та ефективності, а також визначення факторів, що впливають на результати класифікації, наприклад, розмір та склад даних. Результати дослідження можуть бути корисними для розробки нових методів розпізнавання синтетичних текстів, а також для покращення існуючих методів класифікації текстів.

ABSTRACT

Explanatory note: 53 p., 13 fig, 1 tabl., 1 form., 1 ann., 27 sources.

ARTIFICIAL NEURAL NETWORKS, CLASSIFICATION, LANGUAGE PROCESSING, MACHINE LEARNING, SYNTHETIC TEXTS.

The object of research is machine learning methods for detecting synthetic and human-written texts.

The subject of the study is classification methods using machine learning.

The purpose of the study is to identify the most effective methods for recognizing synthetic texts.

In the qualification work, the research focuses on machine learning methods for detecting synthetic texts and human-written texts, with the aim of finding the most effective methods for recognizing synthetic texts. Since synthetic texts can be a difficult task to classify, the use of machine learning and artificial intelligence algorithms can help solve this problem.

The research may include comparing different classification methods and algorithms, evaluating their accuracy and efficiency, and identifying factors that affect classification results, such as the size and composition of the data. The results of the study can be useful for developing new methods for recognizing synthetic texts, as well as for improving existing text classification methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ШНМ – штучна нейрона мережа;

CNN – Convolutional Neural Network – згорткова нейрона мережа;

GAN – Generative Adversarial Network – генеративно змагальна мережа;

NLP – Natural Language Processing – автоматизована обробка природної мови;

RNN – Recurrent Neural Network – рекурентна нейрона мереж.

ВСТУП

Зараз у світі ми багато часу проводимо в Інтернеті, де ми взаємодіємо зі змістом, що складається зі словесної інформації, зображень та відео. Більшість з нас не замислюється про те, які дані є справжніми, а які створені штучно. Синтетичні текстові дані стали надзвичайно популярними в останні роки, особливо з появою штучного інтелекту та машинного навчання.

Зростаюча важливість великих мовних моделей, таких як GPT-4 та ChatGPT, призвела до збільшення обурення щодо академічної доброчесності через потенціал генерації машинного контенту та перефразування. Хоча були проведені дослідження щодо виявлення перефразованого контенту людиною та машинами, порівняння цих типів контенту залишається малодослідженим. З'явилася необхідність у розробці методів виявлення синтетичних текстів, щоби відрізняти їх від реальних даних.

Синтетичні тексти [1] – це тексти, що були згенеровані комп'ютером з використанням різноманітних алгоритмів, наприклад, з використанням машинного навчання або нейронних мереж. Ці тексти можуть бути створені з метою обману, що робить їх особливо небезпечними в контексті дезінформації та маніпулювання громадською думкою.

Виявлення парафраз відіграє вирішальну роль у збереженні цілісності наукових робіт і письмового контенту загалом. Здатність ідентифікувати семантично схожі тексти, незважаючи на значні відмінності у виборі слів і структурі, має важливе значення для різних застосувань, зокрема для виявлення плагіату та оцінки розбіжностей між машинними та людськими переказами. Зростаюча популярність великих мовних моделей, таких як GPT-4 та ChatGPT [2], [3], уможливило автоматичне генерування високоякісних переказів, що ще більше підкреслює потребу в надійних методах виявлення.

Відчутно бракує досліджень, які б порівнювали перекази, створені людиною, з переказами, зробленими машинами. Глибше розуміння їхніх подібностей і відмінностей розуміння їхньої схожості та відмінностей має вирішальне значення для вдосконалення методів виявлення синтетичних текстів та зменшення потенційних загроз академічній доброчесності.

Дослідження семантичних відмінностей між різними текстами є перспективним для поглиблення нашого розуміння машинних і людських текстів. Дослідження патернів, які машини можуть генерувати ефективніше, ніж люди, може стати основою для стратегій доповнення даних, що може бути особливо цінним з огляду на обмежену кількість позитивних навчальних приклад для систем виявлення плагіату.

Розпізнавання випадків, коли машинні тексти дуже схожі на оригінали, написані людиною, дає змогу вивчати семантичні репрезентації у великих мовних моделях, що може призвести до покращення різноманітних завдань NLP [4], включаючи узагальнення тексту та аналіз настроїв.

Важливість розрізнення текстів, написаних людиною та машиною, полягає в тому, що це може допомогти в покращенні роботи з даними, забезпеченні більш точного та достовірного аналізу, а також вдосконаленні автоматичних систем, що обробляють тексти.

1 АНАЛІЗ ПРЕД МЕТНОЇ ОБЛАСТІ

1.1 Обробка природної мови та її завдання

Завдання обробки мови стосуються різних завдань, які можна виконати над текстом природною мовою за допомогою методів обробки природної мови (NLP) [4]. Ці завдання варіюються від простих, таких як токенізація, виділення частин мови та тегування, до більш складних, таких як класифікація тексту, аналіз настрою та генерація тексту.

Генерація тексту, зокрема, є складним завданням у NLP [4], [5], [6], яке передбачає створення тексту, схожого на людський, на основі заданих вхідних даних або контексту. Це можна зробити за допомогою різних методів і технік [1], [7], деякі з яких включають:

– методи на основі правил – ці методи використовують набір заздалегідь визначених правил і шаблонів для створення тексту. Наприклад, система генерації тексту на основі правил може використовувати набір граматичних правил для створення речень на основі вхідних даних;

– методи на основі шаблонів – ці методи використовують заздалегідь визначені шаблони для створення тексту. Наприклад, система генерації тексту на основі шаблонів може використовувати набір шаблонів речень для створення тексту на основі вхідних даних;

– методи на основі машинного навчання [7] – ці методи використовують алгоритми машинного навчання для вивчення шаблонів і взаємозв'язків у текстових даних, а потім використовують ці знання для створення нового тексту. Наприклад, система генерації тексту на основі машинного навчання може використовувати рекурентну нейронну мережу (RNN) для генерації тексту на основі заданих вхідних даних;

– генеративні змагальні мережі (GAN) [7] – ці методи використовують пару нейронних мереж для створення нового тексту. Одна мережа, яка називається генератором, генерує текст на основі заданих

вхідних даних, тоді як інша мережа, яка називається дискримінатором, намагається відрізнити реальний текст від згенерованого. Цей підхід виявився успішним у створенні високоякісного тексту, який важко відрізнити від тексту, написаного людиною.

На додаток до цих методів, існує ще кілька прийомів, які можна використовувати для покращення якості та точності згенерованого тексту. До них належать використання попередньо навчених мовних моделей, точне налаштування моделей під конкретні завдання, а також використання таких методів, як променевий пошук і вибірка top-k [4], [5] для керування процесом генерації.

Загалом, генерація тексту - це захоплююча галузь досліджень у NLP, яка має багато застосувань, зокрема чат-боти, віртуальні асистенти та генерація контенту. З постійним розвитком нових методів і технік якість і точність згенерованого тексту, ймовірно, продовжуватиме покращуватися, що робить його все більш цінним інструментом як для бізнесу, так і для приватних осіб.

Однак створення якісного тексту, що звучить природно, залишається складним завданням у NLP. Однією з головних проблем при створенні тексту є підтримка зв'язності та послідовності в створеному тексті. Це означає, що згенерований текст має сенс у контексті вхідних даних, а також відповідає граматичним правилам і конвенціям.

Ще одним викликом при створенні тексту є уникнення створення упередженого або образливого тексту. Це може бути особливо складно, коли текст генерується на основі даних, що містять упередження, наприклад, гендерні або расові упередження. Тому важливо розробити методи виявлення та пом'якшення упередженості в тексті, що генерується.

Незважаючи на ці виклики, генерація тексту має багато практичних застосувань. Наприклад, його можна використовувати для створення персоналізованих рекомендацій щодо продуктів, для написання креативних

текстів або для створення чат-ботів, які можуть взаємодіяти з користувачами в природній і розмовній манері.

Одним із найперспективніших напрямків досліджень у галузі генерації тексту є використання великих попередньо навчених мовних моделей, таких як GPT-3 та GPT-4 [3], [8], для генерації тексту. Ці моделі були навчені на величезних обсягах текстових даних і можуть генерувати високоякісний текст, який часто важко відрізнити від тексту, написаного людиною.

Отже, генерація тексту – це складна і відповідальна задача в NLP, яка має багато практичних застосувань. Розробка нових методів і прийомів, таких як попередньо навчені мовні моделі та GAN, ймовірно, продовжить покращувати якість і точність згенерованого тексту. Однак важливо пам'ятати про етичні наслідки генерування тексту і розробляти методи виявлення та пом'якшення упередженості у створеному тексті.

Багато проблем у сфері обробки природної мови мають виклики. На жаль, хоча моделі машинного навчання добре виконують багато завдань, їхні результати часто не відповідають людському розумінню. Моделі можуть розпізнавати шаблони у тексті, але вони не можуть повністю зрозуміти його значення.

Хоча ця проблема переходить у сферу філософії, все ж є кілька практичних проблем, які можна вирішити. Одна з них - обробка багатьох мов без втрати змісту тексту під час перекладу. Переклад, будь він машинним або людським, часто не може передати всі особливості вихідного тексту. Інша проблема - зрозуміння змісту кількох документів, що можуть мати різні точки зору, алгоритмів поєднання їх в єдину картину складніше зробити.

І ще одна проблема полягає у зборі даних [1]. Для деяких мов, наприклад англійської, збір даних не є великою проблемою, адже існує багато відповідних наборів даних для NLP. Однак, для менш розповсюджених мов збір даних стає великою проблемою. І хоча деякі

набори даних охоплюють кілька мов, такі як mBERT, який використовує набори даних на основі статей Вікіпедії, часто ці набори даних не охоплюють всі потрібні мови.

1.2 Моделі GPT

Ще кілька років тому, генерація контенту розглядалася з обережністю, оскільки всю увагу зосереджували на моделях word2vec та BERT[3], які відігравали головну роль у сфері обробки природної мови. Але у 2022 році все змінилося, і цей рік ввійшов в історію як початок ери генерації контенту. Значну роль у цьому зіграла компанія OpenAI, яка за 4 роки створила кілька мовних моделей, що показали світу, що розвиток штучного інтелекту просувається швидше, ніж можна було очікувати.

Моделі GPT (Generative Pre-trained Transformer) [2], [7] – це сімейство мовних моделей, які використовують методи глибокого навчання для генерації тексту, схожого на людський. Ці моделі засновані на архітектурі Transformer і попередньо навчаються на великих обсягах текстових даних, що дозволяє їм генерувати зв'язний текст, який звучить природно.

Перша модель GPT, GPT-1, була розроблена OpenAI у 2018 році і мала 117 мільйонів параметрів. Відтоді було випущено кілька потужніших версій, зокрема GPT-2, GPT-3 та GPT-4. Ці моделі навчалися на все більших і різноманітніших наборах даних і тепер мають мільярди параметрів.

Моделі GPT [3] навчаються за допомогою методу, який називається «навчання без нагляду», що передбачає передбачення наступного слова в реченні на основі попередніх слів. Цей процес повторюється для мільйонів речень, що дозволяє моделі вивчати статистичні закономірності та зв'язки між словами в природній мові.

Однією з найбільш вражаючих особливостей моделей GPT є їхня здатність виконувати широкий спектр завдань на природній мові, таких як завершення тексту, переклад, узагальнення, відповіді на запитання і навіть

творче письмо. Ці моделі також можна точно налаштувати на виконання конкретних завдань, надавши їм марковані приклади текстових даних, що дозволяє їм вивчити конкретні шаблони і взаємозв'язки, необхідні для виконання цього завдання.

Моделі GPT використовуються в різних додатках, зокрема в чат-ботах, віртуальних помічниках, створенні контенту тощо. Однак вони також викликають занепокоєння щодо їхнього потенціалу для поширення дезінформації та фейкових новин, а також щодо їхнього впливу на приватність та етичні міркування.

Ще однією помітною особливістю моделей GPT [2] є їхня здатність генерувати високоякісний текст, який важко відрізнити від тексту, написаного людиною. Це викликало занепокоєння щодо потенційного зловживання GPT-моделями для створення фейкових новин, пропаганди або інших форм дезінформації. Щоб вирішити ці проблеми, багато дослідників працюють над розробкою методів виявлення та зменшення потенційної шкоди, яку завдають ці моделі.

Ще однією проблемою, пов'язаною з моделями GPT [7], [8], [9], є їхні високі обчислювальні вимоги та енергоспоживання. Найбільші GPT-моделі мають мільярди параметрів, і їхнє навчання вимагає величезних обсягів даних і обчислювальних потужностей. Це спричинило занепокоєння щодо впливу GPT-моделей на навколишнє середовище та необхідність розробки більш енергоефективних алгоритмів та обладнання.

Незважаючи на ці побоювання, GPT-моделі стають дедалі популярнішими у спільноті обробки природної мови (NLP) і демонструють чудову продуктивність у широкому діапазоні мовних завдань. Вони також надихнули на розробку нових моделей і методів, таких як GShard, T5 і GPT-Neo [3], [8], які спрямовані на усунення деяких обмежень оригінальних GPT-моделей.

Отже, GPT-моделі – це сімейство мовних моделей, які використовують методи глибокого навчання для створення тексту, схожого

на людський. Вони продемонстрували чудову продуктивність у широкому спектрі мовних завдань і мають потенціал для революції в галузі обробки природної мови. Однак вони також піднімають важливі етичні, екологічні та технічні проблеми, які необхідно вирішити, щоб забезпечити їх відповідальне та корисне використання.

2 НАУКОВО-ТЕХНІЧНА ЗАДАЧА ТА ЦІЛІ ДОСЛІДЖЕННЯ

Проблема виявлення синтетичних текстів стає все більш актуальною в сучасному суспільстві, оскільки синтетичні тексти використовуються для поширення дезінформації, пропаганди та іншого шкідливого контенту. Основна проблема полягає в тому, що синтетичні тексти розроблені таким чином, щоб імітувати стиль і тон текстів [9], [10], [11], написаних людиною, що робить їх важко відрізнити від автентичних текстів.

Класифікація текстів є одним з методів виявлення синтетичних текстів. Цей метод включає в себе застосування алгоритмів та моделей машинного навчання для автоматичної обробки та аналізу текстових даних з метою виявлення ознак, які свідчать про синтетичний характер тексту.

Зазвичай, такі ознаки можуть включати в себе використання повторюваних фраз, незвичайні мовні конструкції, неправильну граматику та інші відмінності від стандартних природних мовних зразків. Застосування NLP-методів дозволяє автоматично виявляти такі ознаки та забезпечує ефективний та точний процес виявлення синтетичних текстівх [10].

Постановка проблеми для цього дослідження полягає у визначенні ефективних методів класифікації текстів та розробці найкращих практик для оцінки та порівняння цих методів.

Для вирішення цієї проблеми в дослідженні буде застосовано системний підхід, який передбачає кілька етапів:

– збір та аналіз даних. Необхідно зібрати велику кількість текстових даних, що містять як синтетичні тексти, створені з використанням моделей, таких як GPT, так і натуральні тексти. Далі необхідно провести аналіз цих даних, щоб визначити основні характеристики та властивості синтетичних текстів, такі як структура, стилістичні особливості, лексичні та синтаксичні властивості тощо;

– реалізація методів. На цьому етапі буде реалізовано методи виявлення, які ґрунтуватимуться на низці методів машинного навчання та обробки природної мови. Ці методи будуть призначені для виявлення ключових особливостей синтетичних текстів, які відрізняють їх від автентичних текстів;

– експериментальне порівняння методів. Методи виявлення синтетичних текстів необхідно перевірити на різних наборах даних, щоб оцінити їх ефективність та точність. На цьому етапі можна порівняти різні методи виявлення та визначити найкращий з них.

Загалом це дослідження має на меті проаналізувати ефективні методи виявлення синтетичних текстів і надати найкращі практики для оцінки та порівняння цих методів. Вирішуючи проблему виявлення синтетичних текстів, це дослідження може допомогти пом'якшити шкідливий вплив синтетичних текстів і сприяти формуванню більш проінформованого суспільства, якому можна довіряти.

3 АНАЛІЗ МЕТОДІВ ТА ПІДХОДІВ ДО КЛАСИФІКАЦІЇ ТЕКСТІВ

3.1. Методи векторизації

One-Hot encoding (Count vectorizing) – метод, при якому слова замінюються векторами 0 та 1. Мета полягає в тому, щоб взяти корпус (важливий обсяг слів) і створити вектор кожного унікального слова, що міститься в корпусі. Потім кожне слово буде спроектовано на цей вектор, де 0 означає, що його не існує, а 1 означає, що воно існує.

Term Frequency-Inverse Document Frequency (TF-IDF) [1], [12], [13] – це вага, яка часто використовується в інформаційному пошуку та текстовому майнінгу. Ця вага є статистичним показником, який використовується для оцінки важливості слова для документа в колекції або корпусі.

Цей метод є ефективним, коли йдеться про значну кількість стоп-слів (цей тип слів не є релевантним для інформації). Термін IDF дозволяє виявити важливі та рідкісні слова.

TF-IDF можна використовувати на рівні слів, а також на рівні n-грамів символів. Розглянемо декілька видів TF-IDF векторизації на рівні слова та символів:

– Word-level TF-IDF: це найпростіший вид TF-IDF, який враховує тільки окремі слова в документі. В цьому випадку, кожен документ розбивається на слова (токени), а потім розраховується TF-IDF для кожного слова окремо. Це дає змогу відрізняти корисні слова від загальних, але може бути неефективним для документів з багатьма варіантами написання слів, орфографічними помилками та синонімами;

– N-gram TF-IDF: цей вид TF-IDF розглядає послідовності слів фіксованої довжини n як окремі "слова" і розраховує їх TF-IDF вектори. Наприклад, якщо n = 2, то послідовні пари слів у документі будуть вважатися окремими словами для розрахунку TF-IDF. Це дає змогу

враховувати контекст та послідовність слів у документах та дозволяє знаходити більш точні результати в порівнянні з word-level TF-IDF;

– Symbol-level n-gram TF-IDF: цей вид TF-IDF розглядає послідовності символів фіксованої довжини n як окремі "слова" і розраховує їх TF-IDF вектори. Це дає змогу враховувати морфологію слів, тобто форму та закінчення слів, що може бути корисним для аналізу мови та машинного перекладу. Але цей підхід може призвести до збільшення розміру векторів та складнощів у обробці великих даних.

Попередньо навчена модель FastText [10], [13] – це безкоштовна, легка бібліотека з відкритим вихідним кодом, яка дозволяє користувачам вивчати представлення тексту і текстові класифікатори. Вона працює на стандартному, типовому обладнанні. Моделі можуть бути зменшені в розмірі, щоб поміститися навіть на мобільних пристроях. Це оновлений варіант Word2Vec, який представляє кожен мішок як n-грами символів.

Word Embeddings або Word vectors (WE) [10], [13] – іншим популярним і потужним способом зв'язати вектор зі словом є використання щільних «векторів слів», які також називають «вбудовуваннями слів». У той час як вектори, отримані за допомогою однократного кодування, є двійковими, розрідженими (здебільшого складаються з нулів) і дуже високорозмірними (розмірність дорівнює кількості слів у словнику), «вставки слів» є низькорозмірними векторами з плаваючою комою (тобто «щільними» векторами, на відміну від розріджених векторів). На відміну від векторів слів, отриманих за допомогою однократного кодування, вставки слів визначаються з даних. Часто зустрічаються 256-мірні, 512-мірні або 1024-мірні вставки слів, коли мова йде про дуже великі словники. З іншого боку, однократне кодування слів, як правило, призводить до векторів, які мають розмірність 20 000 або більше (у цьому випадку ми маємо справу зі словником у 20 000 лексем). Таким чином, вставки слів упаковують більше інформації в набагато меншій кількості вимірів.

3.2. Вибір моделі

Вибір моделі [14] – це процес вибору між різними підходами до машинного навчання. Коротше кажучи, різними моделями.

Набір даних зазвичай розділяють на навчальну та тестову частини (валідаційний набір буде визначено в моделях глибокого навчання).

Для класифікації ми будемо використовувати терміни [15], що зображено на рисунку 3.1:

– TP – істинно позитивний прогноз;

– TN – істинно негативний прогноз;

– FP – хибнопозитивний прогноз;

– FN – хибнонегативний прогноз.

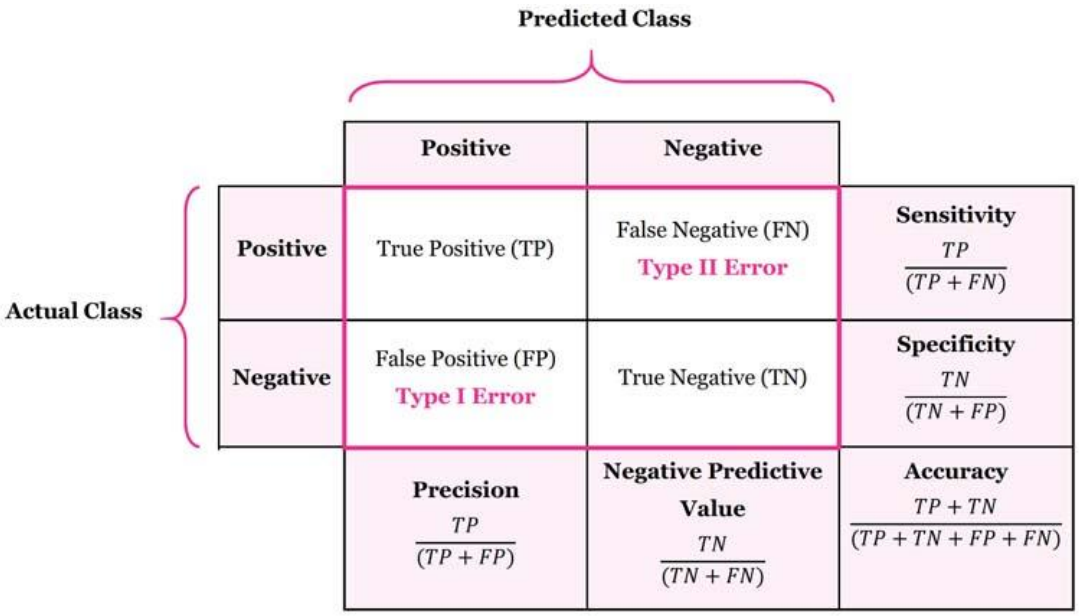

Рисунок 3.1 – Confusion matrix

Accuracy (точність) – всі позитивні прогнози на всіх прогнозах.

Balanced Accuracy (збалансована точність) – визначається, як середнє значення відгуку, отриманого для кожного класу для незбалансованого набору даних.

Precision (Точність) – це інтуїтивно зрозуміла здатність класифікатора не позначати, як позитивний зразок, який є негативним.

Recall або sensitivity (відтворення або чутливість) – це інтуїтивно зрозуміла здатність класифікатора знаходити всі позитивні зразки.

f1-score – оцінка f1 може бути інтерпретована як середньозважене значення точності та відгуку, що наведено у формулі 3.1:

$$f1 = \frac{2 * (\text{precision} * \text{recall})}{\text{precision} + \text{recall}}, \quad (3.1)$$

де $\text{precision} = \frac{TP}{TP+FP}$;

$\text{recall} = \frac{TP}{TP+FN}$;

TP – істинно позитивний прогноз;

FP – хибнопозитивний прогноз;

FN – хибнонегативний прогноз.

Cohen kappa – це оцінка, яка виражає рівень згоди між двома анотаторами щодо проблеми класифікації:

Рекомендаційна система має виконувати наступний перелік задач:

– 0,4 – це досить погано;

– від 0,4 до 0,6 – еквівалентно людському;

– від 0,6 до 0,8 – дуже добре;

– більше 0,8 – винятково добре.

Коефіцієнт кореляції Метьюса (Matthews correlation coefficient) – використовується в машинному навчанні як міра якості бінарних (двокласових) класифікацій. Коефіцієнт враховує істинні та хибні позитивні та негативні результати і зазвичай вважається збалансованою мірою, яку можна використовувати, навіть якщо класи мають дуже різні розміри. MCC - це, по суті, коефіцієнт кореляції між спостережуваною та прогнозованою бінарними класифікаціями; він повертає значення від -1 до +1:

– +1 – означає ідеальне передбачення;

– 0 – не краще, ніж випадкове передбачення;

– -1 – вказує на повну розбіжність між передбаченням і спостереженням.

3.3 Cross-validation

Для того, щоб реалізувати надійне порівняння між моделями, потрібно перевірити надійність кожної моделі.

Перехресна перевірка (cross-validation) [14], [15] – це широко використовувана техніка в машинному навчанні та науці про дані для оцінки продуктивності моделі. Основна ідея перехресної перевірки полягає в тому, щоб розбити наявні дані на кілька підмножин, або складень, і використовувати кожне складення по черзі як валідаційний набір під час навчання моделі на решті даних. Це дозволяє перевірити роботу моделі на декількох підмножинах даних, і таким чином отримати більш точну оцінку її узагальнюючої здатності. Перехресна перевірка особливо корисна в ситуаціях, коли наявні дані обмежені, зашумлені або незбалансовані, оскільки вона допомагає зменшити ризик надмірного пристосування і дає більш реалістичну оцінку справжньої продуктивності моделі. Існує кілька типів методів перехресної перевірки, включаючи k-кратну перехресну перевірку, перехресну перевірку з виключенням одного та стратифіковану перехресну перевірку, кожен з яких має свої сильні та слабкі сторони. Загалом, перехресна перевірка є важливим інструментом в інструментарії машинного навчання та науки про дані і широко використовується на практиці для забезпечення точності та надійності моделей.

При k-кратній (k-fold) [15] перехресній перевірці, що зображено на рисунку 3.2, ми спочатку розбиваємо наш набір даних на k однакових за розміром підмножин. Потім ми повторюємо метод навчання-тестування k разів так, що кожного разу одна з k підмножин використовується як тестова

множина, а решта k-1 підмножин використовуються разом як навчальна множина. Нарешті, ми обчислюємо оцінку продуктивності моделі шляхом усереднення оцінок за k тестів.

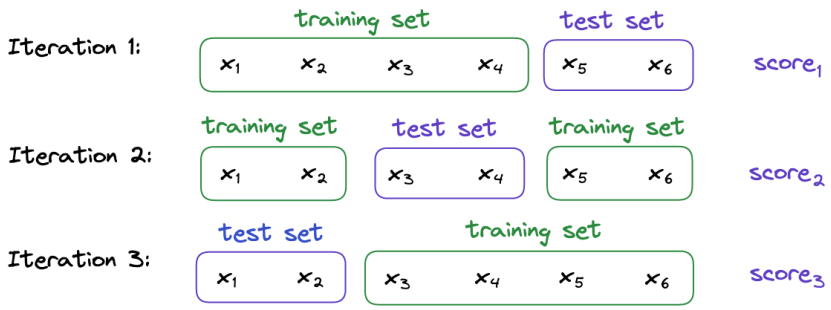

Рисунок 3.2 – K-fold validation

При перехресній перевірці методом пропущених даних (leave-one-out, LOO) [15], що зображено на рисунку 3.3 нижче, ми навчаємо нашу модель машинного навчання n разів, де n – дорівнює розміру нашого набору даних. Кожного разу лише одна вибірка використовується як тестовий набір, тоді як решта використовується для навчання нашої моделі.

Рисунок 3.3 – Leave-one-out validation

Мета полягає в тому, щоб обчислити кожну метрику для кожного спліту і обчислити їх середнє значення (mean) і стандартне відхилення (std).

3.4 Моделі для класифікації

Переглянемо моделі машинного навчання, моделі глибокого навчання та спеціалізовані моделі NLP.

3.4.1 Наївний Байєс

Наївний Байєс – це імовірнісна модель, яка використовує теорему Байєса для класифікації даних на основі частоти появи кожної ознаки. Зазвичай використовується в задачах обробки природної мови, таких як класифікація текстів. Він заснований на теоремі Байєса і припускає, що поява кожної ознаки (або атрибута) не залежить від інших ознак, заданих міткою класу. Це припущення часто є нереалістичним, але воно дозволяє алгоритму бути обчислювально ефективним і простим у реалізації.

Наївний Байєс [15], [16] працює, спочатку вивчаючи ймовірності того, що кожна ознака (або атрибут) потрапляє до кожного класу, враховуючи навчальний набір даних. Потім, коли з'являється новий екземпляр (з власним набором ознак), він обчислює ймовірність належності цього екземпляра до кожного класу, перемножуючи ймовірності окремих ознак, що зустрічаються в цьому екземплярі, з урахуванням ймовірностей класів, вивчених під час навчання. Клас з найвищою ймовірністю потім призначається як передбачуваний клас для даного екземпляра.

Наївний Байєс особливо корисний для завдань класифікації тексту оскільки він може обробляти великі обсяги текстових даних і відносно швидко навчається. Він добре працює для таких завдань, як виявлення спаму, аналіз настроїв і класифікація тем. Однак, одним з обмежень є те, що він припускає, що ознаки є незалежними одна від одної, що не завжди може

бути правдою в реальних даних. Тим не менш, наївний Байєс – це простий і ефективний алгоритм, який може бути хорошим вибором для багатьох завдань класифікації.

3.4.2 Логістична регресія

Логістична регресія – це алгоритм машинного навчання, який широко використовується для задач бінарної класифікації. Він працює шляхом моделювання ймовірності бінарного результату (наприклад, «так» чи «ні», «правда» чи «брехня») за допомогою логістичної функції, також відомої як сигмоїдна функція. Результатом логістичної функції є значення між 0 і 1, яке можна інтерпретувати як ймовірність позитивного класу (наприклад, так, істина).

Логістична регресія працює [14], спочатку вивчаючи вагу кожної ознаки (або атрибута), яка впливає на ймовірність позитивного класу, на основі навчального набору даних. Потім, коли з'являється новий екземпляр (з власним набором ознак), вона обчислює ймовірність позитивного класу, використовуючи вивчені ваги та значення ознак цього екземпляра. Якщо ймовірність більша за порогове значення (зазвичай 0,5), позитивний клас призначається як передбачуваний клас для екземпляра. В іншому випадку призначається негативний клас.

Логістична регресія широко використовується в задачах бінарної класифікації, таких як виявлення шахрайства, прогнозування відтоку клієнтів і медична діагностика. Її легко інтерпретувати і вона може працювати як з числовими, так і з категоріальними даними. Однак він припускає, що зв'язок між ознаками та результатом є лінійним, що не завжди відповідає дійсності в реальних даних. На практиці часто використовують методи регуляризації, такі як регуляризація L1 або L2, щоб запобігти надмірному пристосуванню та підвищити продуктивність [20].

3.4.3 Машини опорних векторів

Машина опорних векторів (SVM) – це алгоритм машинного навчання, який широко використовується для завдань класифікації, регресії та виявлення викидів. У контексті класифікації SVM працює, знаходячи гіперплощину, яка найкраще розділяє точки даних різних класів в високовимірному просторі ознак. Гіперплощина вибирається таким чином, щоб максимізувати відстань між двома найближчими точками різних класів, відомими як опорні вектори.

SVM може обробляти як лінійно роздільні, так і нелінійно роздільні дані [21], використовуючи функцію ядра, яка відображає вхідні дані в вищорозмірний простір ознак, де вони можуть стати роздільними. Розповсюдженими функціями ядра є лінійна, поліноміальна, радіально-базисна функція (RBF) та сигмоїдальна.

SVM широко використовується в застосуваннях, таких як класифікація зображень, біоінформатика та обробка природних мов. У нього є декілька переваг порівняно з іншими алгоритмами класифікації, такими як здатність обробляти високорозмірні дані, здатність обробляти нелінійно роздільні дані та здатність уникнути перенавчання. Однак SVM може страждати від високої обчислювальної складності та чутливості до вибору гіперпараметрів [15], таких як функція ядра та її параметри. На практиці часто використовуються техніки, такі як пошук по сітці та крос-валідація, щоб знайти оптимальні гіперпараметри.

3.4.4 Стохастичний градієнтний спуск

Стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD) – це алгоритм оптимізації, який використовується в машинному навчанні для зменшення функції втрат при тренуванні моделі. Він є розширенням

класичного градієнтного спуску, який використовується для знаходження мінімуму функції шляхом ітеративного зменшення значення.

Одна з особливостей SGD [20] полягає в тому, що він використовує випадковий вибір підмножини даних для кожної ітерації. Це означає, що в кожній ітерації SGD використовує лише частину даних для обчислення градієнту функції втрат. Це дозволяє виконувати більше ітерацій за той же час, що дозволяє знайти мінімум функції швидше.

Одним з головних переваг SGD [4] є його ефективність в роботі з великими наборами даних, оскільки він дозволяє швидко знаходити мінімум функції втрат навіть при великій кількості даних. Крім того, SGD дозволяє уникнути локальних мінімумів, які можуть бути знайдені класичним градієнтним спуском.

Однак, існують деякі недоліки SGD. Оскільки він використовує випадковий вибір підмножини даних для кожної ітерації, це може призвести до шуму в оцінках градієнту, особливо при невеликих розмірах підмножини даних. Крім того, SGD може бути чутливим до початкових умов і параметрів алгоритму.

3.4.5 K-Nearest-Neighbors

k-Nearest Neighbors (k-NN) – це алгоритм навчання з нагляду, який використовується в машинному навчанні для класифікації та регресії даних. В основі k-NN лежить ідея, що подібні дані повинні бути більш ймовірно віднесені до одного класу або мати схожі значення.

Концепція k-NN [15] полягає в тому, щоб знайти k найближчих сусідів до нового невідомого зразка даних, використовуючи відстань між зразками даних. Потім для класифікації цього зразка даних, k-NN визначає більшість класів серед його k найближчих сусідів, і визначає, що новий зразок належить до цього класу. Якщо кількість найближчих сусідів (k) дорівнює 1, то алгоритм використовується як найближчий сусід (1-NN).

У випадку регресії, замість класів, k-NN [20] використовує середнє значення або медіану значень залежно від значень змінних вихідної змінної у к найближчих сусідів.

Однією з головних переваг k-NN є його простота та інтерпретованість. Крім того, цей алгоритм добре показує себе у випадку невеликих наборів даних, а також у випадку нелінійних проблем класифікації та регресії.

Однак, k-NN може показувати низьку точність у випадку великих наборів даних, оскільки збільшення кількості зразків даних може призвести до збільшення часу обчислень, необхідних для виконання алгоритму. Крім того, кількість найближчих сусідів (k) є параметром, який може вплинути на точність класифікації та регресії. Якщо значення k занадто мале, то це може призвести до перенавчання моделі, тоді як занадто велике значення k може призвести до недооптимізації моделі [20].

Існує також питання вибору метрики відстані між зразками даних, яка визначає, наскільки близькі два зразки. Найчастіше використовується евклідова відстань, але можуть бути використані інші метрики відстані, такі як косинусна відстань чи Манхеттенська відстань.

Щоб покращити результати k-NN, можуть бути використані різні техніки, такі як нормалізація даних [20], зменшення кількості ознак, використання ваги сусідів та зменшення розмірності даних за допомогою методів зменшення розмірності.

У загальному, k-NN є простим та ефективним алгоритмом, який добре показує себе у випадку невеликих наборів даних з чіткою розмежуваною структурою. Це також дуже популярний метод для використання у задачах класифікації та регресії в машинному навчанні.

3.4.6 Випадковий ліс

Випадковий ліс класифікатор – це алгоритм машинного навчання, який використовується для задач класифікації, регресії та кластеризації

даних. Він є одним з найбільш популярних алгоритмів машинного навчання завдяки своїй високій точності та швидкодії.

Основна ідея випадкового лісу полягає в створенні багатьох дерев рішень, які називаються «деревами рішень», і об'єднанні їх результатів для забезпечення кращої точності класифікації. Кожне дерево рішень випадкового лісу навчається на підмножині вхідних даних, яку можна отримати за допомогою методу «бутстрапа» [14] – заміни кожного елементу вибірки даних на рандомні значення з деякого розподілу. Кожне дерево рішень обчислює прогнози для кожного зразка даних та результати голосування використовуються для визначення остаточного класифікаційного рішення.

Важливою особливістю випадкового лісу є використання випадкових функцій для визначення поділу дерев [14]. Наприклад, для кожного поділу вибірки можна випадково обрати підмножину функцій, які будуть використовуватися для прийняття рішень, що допомагає уникнути перенавчання та підвищити загальну точність.

Випадковий ліс класифікатор також може бути використаний для визначення важливості функцій, що допомагає виявити найбільш важливі фактори в задачі класифікації або регресії [14].

Використання випадкового лісу класифікатора має деякі переваги порівняно з іншими алгоритмами машинного навчання. Перш за все, він є дуже ефективним, що дозволяє обробляти великі обсяги даних в короткий проміжок часу. Крім того, випадковий ліс класифікатор не вимагає складного налаштування гіперпараметрів, які можуть бути складні для встановлення в інших алгоритмах.

Однак, випадковий ліс класифікатор також має свої недоліки. Наприклад, він може бути менш точним порівняно з більш складними алгоритмами, особливо якщо дані містять багато шуму. Крім того, випадковий ліс може мати проблему з уникненням перенавчання, якщо використовується надто велика кількість дерев.

У випадку задачі класифікації, випадковий ліс класифікатор може бути використаний для вирішення різних проблем, таких як визначення відповідності документів або виявлення шахрайських транзакцій [19]. Крім того, випадковий ліс класифікатор може бути використаний для різних видів задач регресії, наприклад, для прогнозування цін на нерухомість або вартості акцій.

У загальному, випадковий ліс класифікатор є потужним та ефективним алгоритмом машинного навчання, який знаходить застосування в багатьох галузях науки та бізнесу.

3.4.7 XGBoost

XGBoost (Extreme Gradient Boosting) є потужним алгоритмом машинного навчання, який може використовуватися як для задач класифікації, так і для регресії. Він є покращенням традиційного градієнтного бустингу та має кілька переваг.

Однією з ключових переваг XGBoost є його ефективність. Алгоритм може працювати з великими обсягами даних, включаючи дані з багатьма ознаками, і швидко вивчати складні залежності між ознаками. Це дозволяє отримати високу точність прогнозування відповідей, а також зменшити час, потрібний для навчання моделі.

Іншою важливою перевагою XGBoost є можливість регуляризації. Алгоритм використовує L1 та L2 регуляризацію для контролю ваг ознак та попередження перенавчання. Це дозволяє моделі стати більш універсальними та зменшити кількість ознак, які не впливають на результат.

Крім того, XGBoost використовує багатоступінчастий процес оптимізації, що дозволяє моделі швидко знаходити глобальний мінімум функції втрат, що мінімізує помилки прогнозування. Цей підхід дозволяє XGBoost досягти високої точності прогнозування, особливо у порівнянні з іншими алгоритмами машинного навчання.

XGBoost має декілька варіантів, таких як XGBoost-Linear [14], який використовує лінійну модель для зменшення кількості параметрів та поліпшення ефективності. Також, XGBoost можна використовувати для вирішення задач класифікації та регресії в багатокласових сценаріях та використовувати для роботи з різними типами даних, такими як числові, категоріальні та текстові.

Ще однією важливою функцією XGBoost є можливість визначення ваг для різних точок даних. Це дозволяє звернути більше уваги на деякі важливі дані та забезпечити більш точне прогнозування результатів.

Щоб використовувати XGBoost [16], необхідно спочатку навчити модель на тренувальному наборі даних, визначити оптимальні параметри та провести перевірку моделі на тестових даних, щоб переконатися у її точності. Після цього модель може бути використана для прогнозування результатів на нових наборах даних.

Узагалі XGBoost є потужним та ефективним алгоритмом машинного навчання, який забезпечує високу точність прогнозування в різних сценаріях та забезпечує швидкість та ефективність обробки великих обсягів даних. Тому він є популярним інструментом для вирішення різних задач машинного навчання.

3.4.8 Моделі глибокого навчання. Глибока нейронна мережа

Моделі глибокого навчання – це нейронні мережі з декількома шарами, які можуть вивчати складні функції з даних. Ось короткий опис деяких поширених моделей глибокого навчання:

Глибока нейронна мережа (Deep Neural Network, DNN) – це тип нейронної мережі з декількома прихованими шарами. Використовується для керованих завдань навчання, таких як класифікація зображень, розпізнавання мови та обробка природної мови.

3.4.9 Recurrent neural network

Рекурентні нейронні мережі (Recurrent neural network, RNN) – це клас нейронних мереж, які здатні оброблювати послідовності даних різної довжини, зокрема текстові послідовності, часові ряди та інші. Одна з основних властивостей RNN полягає в тому, що вони можуть зберігати інформацію про попередні стани мережі та використовувати цю інформацію для обробки нових вхідних даних.

Основна ідея RNN полягає в тому, щоб передавати стани мережі в кожен момент часу, тобто використовувати інформацію з попередніх часових кроків для обробки поточного вхідного сигналу. Це дозволяє мережі зберігати інформацію про контекст та залежності в даних, що може бути важливим для багатьох задач, наприклад, для аналізу тексту.

RNN складається з набору нейронів, які пов'язані між собою відносинами зворотного зв'язку. Кожен нейрон приймає вхідний сигнал та стан мережі з попереднього часового кроку та відправляє свій вихідний сигнал наступному нейрону в мережі. Завдяки цьому зв'язку RNN можуть запам'ятовувати інформацію про попередні стани [19] мережі та використовувати цю інформацію для обробки нових вхідних даних.

Одним з найбільш популярних застосувань RNN є генерація тексту, де мережа навчається передбачати наступне слово в тексті, враховуючи попередні слова. Також RNN застосовуються для обробки часових рядів, наприклад, для прогнозування цін на акції та інші задачі, де необхідно працювати з послідовністю даних.

Одним з недоліків RNN є те, що при обробці довгих послідовностей вони можуть втратити інформацію з попередніх станів мережі. Це проблема, яку називають проблемою зникновення градієнта. Щоб уникнути цієї проблеми, були розроблені інші типи рекурентних мереж, наприклад, LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) [22].

LSTM та GRU мережі розв'язують проблему зникновення градієнта [20], дозволяючи мережі вибирати, яку інформацію потрібно зберігати та яку інформацію можна ігнорувати. Це досягається за допомогою спеціальних «воріт», які контролюють потік інформації в мережі.

Застосування RNN та їх модифікацій включають розпізнавання мови, машинний переклад, аналіз тексту та часових рядів, створення музики та відео, генерацію тексту та багато іншого.

Завдяки своїй здатності до обробки послідовностей даних, RNN та їх варіанти залишаються важливими інструментами в галузі машинного навчання, особливо в обробці текстової інформації та часових рядів.

3.4.9 Convolutional Neural Network

Згорнуті нейронні мережі (Convolutional Neural Networks або CNN) є основою для багатьох задач комп'ютерного зору, таких як визнання облич, класифікація зображень та розпізнавання об'єктів.

Основними складовими CNN є шари згортки [23], які використовуються для виявлення різних ознак на зображеннях. Кожен шар згортки складається з набору фільтрів, які переміщуються по зображенню та виконують операцію згортки. Ці фільтри допомагають виявляти різні характеристики зображення, такі як границі, кути, текстури тощо.

Після проходження через шари згортки, отримані вектори ознак подаються на повнозв'язний шар, який виконує класифікацію [23]. Також, для покращення точності можуть використовуватись шари підсумування, які зменшують розмір векторів ознак, та шари нормалізації, які забезпечують стабільність навчання.

CNN дозволяють досягти високої точності в розпізнаванні зображень, особливо при наявності великої кількості даних для навчання. Вони також

успішно застосовуються в багатьох галузях, таких як медицина, автомобільна промисловість, робототехніка тощо.

Крім класифікації зображень, CNN можуть бути використані для розв'язання багатьох інших задач, таких як відеоаналітика, обробка мовлення та обробка сигналів. Наприклад, відеоаналітика використовує CNN для визначення рухомих об'єктів на відео, а обробка мовлення використовує CNN для розпізнавання голосу та синтезу мови [24].

Одним з головних викликів в застосуванні CNN є питання обробки зображень з різною роздільною здатністю та розміром. Для розв'язання цього питання використовуються різні методи, такі як збільшення розміру зображення, обрізання та масштабування [20].

У цілому, CNN є потужним інструментом для обробки зображень та вирішення багатьох задач комп'ютерного зору. Застосування цих мереж у багатьох сферах дозволяє отримувати точні та ефективні результати.

3.4.10 Gated Recurrent Unit

Gated Recurrent Unit (GRU) – це тип ШНМ, який схожий на LSTM, але має менше параметрів. Вони часто використовуються в задачах обробки природної мови, таких як розпізнавання мови та переклад.

Основна ідея GRU полягає в тому, щоб замінити стандартний рекурентний блок з двома блоками: reset gate та update gate. Reset gate визначає, яку частину попереднього стану необхідно «забути», а update gate визначає, яку частину нового вхідного сигналу необхідно «запам'ятати» [25]. Це дає можливість контролювати потік інформації, який проходить через мережу.

Окрім цього, GRU має меншу кількість параметрів порівняно зі стандартними рекурентними мережами, такими як Long Short-Term Memory (LSTM), що робить його більш ефективним для навчання на невеликих даних.

GRU був запропонований у 2014 році [20] та став популярним в наукових дослідженнях та реальних застосуваннях, таких як машинний переклад, обробка мовлення, аналіз даних та інші. Він є потужним інструментом для роботи з послідовними даними та досягнення високої точності у багатьох завданнях.

Додатково, GRU має деякі переваги порівняно з LSTM [25]. Зокрема, він може бути більш ефективним у використанні ресурсів, так як має менше параметрів, а також більш швидким у навчанні, що дозволяє працювати з більшими обсягами даних. Крім того, GRU може бути більш зручним у використанні, оскільки не має складної структури з трьома гейтами, як у LSTM.

Одним з застосувань GRU є генерація тексту. Використовуючи GRU, можна навчити модель генерувати нові речення або продовжувати існуючі, враховуючи контекст та розуміння мови. Також GRU можна використовувати для аналізу електроенцефалограм (ЕЕГ) сигналів у медицинській діагностиці, аналізу акційних ринків та інших застосувань, що вимагають обробки послідовних даних [20].

Загалом, GRU є ефективним інструментом для обробки послідовних даних, який дозволяє досягнути високої точності в багатьох завданнях машинного навчання. Його використання стає все більш поширеним у наукових дослідженнях та в реальних застосуваннях, що підтверджує його потужність та ефективність.

3.4.11 Long Short-Term Memory

Long Short-Term Memory (LSTM) – це рекурентна нейронна мережа, що була спеціально розроблена для роботи з послідовними даними, такими як мовлення, музика, текст, часові ряди тощо. LSTM була запропонована у 1997 році Йоханном Гоерем та Фредеріком Гартою і з тих пір стала однією з найпопулярніших моделей у галузі обробки послідовних даних.

LSTM складається з декількох блоків пам'яті, кожен з яких має змінні для зберігання попереднього стану та виводу. Кожен блок має три гейти: вхідний гейт, забуваючий гейт та вихідний гейт. Вхідний гейт визначає, які дані потрібно зберегти, забуваючий гейт визначає, які дані потрібно видалити з попереднього стану, а вихідний гейт визначає, які дані потрібно вивести. Ці гейти дозволяють LSTM довше запам'ятовувати інформацію, що зазвичай є складною задачею для звичайних рекурентних нейронних мереж.

LSTM дозволяє ефективно працювати з даними, що мають довготривалу залежність, тобто де події, що відбуваються далеко від поточного моменту, можуть мати великий вплив на майбутнє. LSTM також здатна автоматично визначати та розуміти рівень важливості даних, що сприяє досягненню більшої точності в багатьох завданнях машинного навчання, таких як машинний переклад, розпізнавання мовлення, генерація тексту та інші.

Загалом, LSTM є потужним та ефективним інструментом для роботи з послідовними даними, який відкриває нові можливості для багатьох завдань у галузі машинного навчання. Проте, використання LSTM також може бути вимогливим до ресурсів, оскільки вона має більшу кількість параметрів, порівняно з іншими моделями, такими як RNN. Тому, при використанні LSTM потрібно враховувати розмір даних та обчислювальні ресурси, що доступні.

Одним з варіантів використання LSTM є комбінування її з іншими моделями машинного навчання, такими як CNN, для отримання кращої точності та швидкості роботи. Наприклад, у галузі обробки зображень, можна використовувати CNN для отримання ознак зображення, а потім використовувати LSTM для аналізу послідовності отриманих ознак та вирішення завдань, таких як розпізнавання мови та генерація тексту.

У загальному, LSTM є потужним та ефективним інструментом для роботи з послідовними даними, який знайшов широке використання в різних галузях. Її можна використовувати для завдань, що потребують

довготривалої залежності та розуміння контексту, таких як машинний переклад, розпізнавання мовлення, аналіз часових рядів та багато інших.

3.4.12 Recurrent Convolutional Neural Network

Recurrent Convolutional Neural Network (RCNN) – це гібридна модель, яка комбінує в собі дві архітектури машинного навчання: Recurrent Neural Network (RNN) та Convolutional Neural Network (CNN).

Ця модель була запропонована для розв'язання завдань, що пов'язані з обробкою послідовних даних, таких як текстові дані, аудіо- та відеодані. RCNN може працювати з даними різної довжини та зберігати контекст з попередніх кроків, що робить її досить потужним інструментом.

За допомогою CNN, RCNN може ефективно використовувати властивості зображень та отримувати розуміння ознак з даних. Після цього, за допомогою RNN, модель може виконувати розуміння послідовності та зберігати контекст з попередніх кроків [25].

Один з прикладів використання RCNN – це відповідь на запитання відносно текстового документу [26]. Модель може отримувати текстовий документ як вхід та виділяти в ньому відповідні ознаки, такі як слова та речення, за допомогою CNN. Потім, за допомогою RNN, модель може аналізувати послідовність та зберігати контекст з попередніх кроків, щоб зрозуміти зміст документа та дати відповідь на запитання.

RCNN є потужним та ефективним інструментом для роботи з послідовними даними, що потребують розуміння контексту та використання властивостей зображень. Вона може бути застосована для різноманітних завдань, таких як розпізнавання мови, машинний переклад, аналіз часових рядів та багато інших.

3.4.13 Трансформери

Трансформер – це тип нейронних мереж, які призначені для обробки послідовних даних, таких як текст і дані часових рядів. Вони широко використовуються в задачах обробки природної мови, таких як переклад і генерація тексту [26], [27]:

– BERT – модель трансформера, яка враховує лівий і правий контексти на всіх рівнях;

– T5 – трансформер передачі тексту в текст, створений для перекладу та подібних завдань.

4 ЕКСПЕРИМЕНТ З ОЦІНЮВАННЯМ МОДЕЛЕЙ

4.1 Створення датасету

Створення датасету на основі творів для ЗНО з української мови для детекції синтетичних творів може бути цікавим завданням для дослідників інформаційної безпеки, які працюють над виявленням фейкових текстів, створених з метою дезінформації або маніпуляції громадською думкою.

Такий датасет може містити тексти українською мовою, які мають власне висловлювання автора, що є дуже показовим для виявлення особливостей тексту написаного людиною та комп’ютером. Після цього можна було б використати алгоритми машинного навчання, щоб створити модель, яка здатна розрізняти синтетичні твори від оригінальних.

Проте, при створенні такого датасету необхідно дотримуватися етичних стандартів, а також дотримуватися законів про захист авторських прав. Крім того, важливо забезпечити права на використання творів, які входять до датасету, та дотримуватися вимог авторського права.

Також необхідно зазначити, що створення такого датасету може стати завданням з високими витратами, які пов'язані зі збором, обробкою та анотуванням великої кількості текстових матеріалів. Проте, якщо такий датасет буде створено з високою якістю, він може бути корисним для багатьох дослідників у різних галузях, які працюють над розвитком інструментів для боротьби з дезінформацією та маніпуляцією громадською думкою.

Тексти, які можна вважати людськими, можна знайти в різних джерелах, включаючи збірники для підготовки до ЗНО. Такі збірники зазвичай містять різні теми, з яких можна взяти власні висловлювання для створення датасету. Наприклад, можна взяти текстові зразки на такі теми, як:

– «як стереотипи впливають на життя людини?»;

– «чому сьогодні толерантність – це яскравий вияв ступеня демократії держави й одна з умов її розвитку?»;

– «які складники креативності, на Вашу думку, є особливо важливими для самореалізації людини та її конкурентоспроможності на ринку праці?».

Ці теми та інші можуть становити основу для створення датасету, що зображено на рисунку 4.1, що містить власні висловлювання, що можна вважати написаними людиною.

	A	B	C
1	id	topic	text
2	1	Як стереотипи впливають на життя людини?	Стереотипи є своєрідним фільтром, через який людина сприймає дійсність. Переконана, що роль стереотипів у житті особистості є суперечливою: з одного боку, вони роблять наше існування більш зрозумілим і впорядкованим, а з іншого – обмежують можливість пізнання нового, сприяють формуванню хибних уявлень та упереджень. По-перше, на основі вироблених стереотипів людина може мати вже сформоване враження, часто необ'єктивне, про іншу особу ще до початку знайомства з нею. Згадую один випадок зі свого шкільного життя. Кілька років тому в нашому класі з'явився новий учень. Це був хлопець, який дуже добре вчився, але з ним ніхто не хотів спілкуватися, адже мої однокласники мали упереджену думку: відмінники – зануди. Стереотип у цій ситуації зумовив помилкове ставлення до особистості й призвів до хибних висновків про хлопця. Лише через кілька місяців ми зрозуміли, що наш однокласник є «душею компанії», він товариський, дотепний і готовий прийти на допомогу кожному, хто цього потребує. По-друге, деякі зі стереотипів передаються з покоління в покоління, їх приймають на віру, навіть
3	2	Чому сьогодні толерантність – це яскравий вияв	Толерантність у суспільстві, мабуть, є одним з найяскравіших показників рівня культури членів
4	3	Які складники креативності, на Вашу думку, є	Сучасний ринок праці є динамічним, змінним і гнучким. Він вимагає від працівників готовності
5	4	Який зі способів подолання конфліктів є найбільш	Життя – це складний процес, упродовж якого постійно виникають проблеми й непорозуміння.
6	5	Як можна бути патріотом, живучи поза межами своєї	Усім відомо, що для кожної людини рідний край найдорожчий і наймиліший серцю, проте
7	6	Що краще в житті: вірити у власні сили й ризикува	Я переконана, що обережність у діях часто заважає нам досягти цілей. Людина здатна
8	7	Наскільки шкідливою є така залежність від оцінки	Сучасна людина не уявляє свого життя без інтернету. Це й не дивно, оскільки саме він дає
9	8	Як же вберегтися від зневіри на шляху до мрії?	У кожної людини є мрія, яку хочеться втілити в життя. Я думаю, що на цьому шляху варто бути
10	9	Наскільки важливим є вміння чесно оцінювати се	Цілком справедливим буде твердження, що вміння чесно оцінювати себе, готовність визнавати
11	10	Чи здатне спілкування он-лайн замінити людині	У сучасному світі живе спілкування стало розкішшю. Переконаний, що віртуальність не може
12	11	Що ж мотивує нас до дії більше – віра в себе чи	Упродовж життя людина ставить перед собою певні цілі, виконує різноманітні складні завдання
13	12	Що краще: зосередитися на певній діяльності й	Упевнена в тому, що сучасна особистість має бути всебічно розвиненою, здатною до
14	13	Наскільки важливим є дотримання правил співжиття	Людина не може існувати відокремлено від соціуму та його впливу. Я думаю, що особистість
15	14	Як варто чинити, зіткнувшись із невихованістю,	Реакція на грубість і невихованість у кожному конкретному випадку залежить від ситуації й
16	15	Як потрібно реагувати на виклики, перед якими нас ст	Жаліти себе – це певною мірою намагатися сховатися від проблем, заховати голову в пісок.
17	16	Як можна з користю проводити вільний час, щоб потім	Здавалося б, наше життя таке швидкоплинне, що люди постійно відчувають дефіцит часу. Тому,

Рисунок 4.1 – Тексти написані людиною

Далі згенеруємо тексти за зібраними темами за допомогою GPT. Для початку, потрібно налаштувати середовище для розробки. Буде використовуватися Google Colab. Для використання OpenAI API [2], [3] необхідно зареєструватися на сайті OpenAI, створити API-ключ та сконфігурувати налаштування запиту.

Щоб звернутися до моделі GPT за допомогою OpenAI API, необхідно виконати наступні кроки:

– створити API-ключ на сайті OpenAI та сконфігурувати налаштування запиту, вказавши параметри, такі як довжина згенерованого тексту та тематика;

– використовувати сторонні бібліотеки для взаємодії з API, наприклад, requests для Python. Після цього можна створити запит до API, як показано на рисунку 4.2, передавши API-ключ та параметри запиту [3];

```python
def generate_chat_completion(messages, model="gpt-3.5-turbo", temperature=1, max_tokens=None):
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {API_KEY}",
    }

    data = {
        "model": model,
        "messages": messages,
        "temperature": temperature,
    }

    if max_tokens is not None:
        data["max_tokens"] = max_tokens

    response = requests.post(API_ENDPOINT, headers=headers, data=json.dumps(data))

    if response.status_code == 200:
        return response.json() #response.json()["choices"][0]["message"]["content"]
    else:
        raise Exception(f"Error {response.status_code}: {response.text}")

```

Рисунок 4.2 – Функція запиту до GPT API

– обробити відповідь API, яка містить згенерований текст. В залежності від бібліотеки та мови програмування, яку ви використовуєте, обробка відповіді може відрізнятися.

Після налаштування середовища для розробки та отримання токену та ключа, можна почати створювати тексти за заданими темами. Для цього спочатку потрібно визначити теми. Після того, як було вибрано тему, можна передати її в модель GPT, як на рисунку 4.3, щоб отримати згенерований текст.

```python
def generate_text(topic):
  request_message = "Напиши висловлюння на тему: '{0}'. " +
  "Обсяг приблизно 250 слів. " +
  "Структура: вступ, приклад з літератури, приклад з життя та висновок."
  messages = [{"role": "user", "content": request_message.format(topic)}]
  response_text = generate_chat_completion(messages, model="gpt-3.5-turbo")
  print(response_text)
  return (response_text, response_text["choices"][0]["message"]["content"])
```

Рисунок 4.3 – Запит з темою до GPT

Було використано наступний запит: «Напиши висловлюння на тему: «{0}». Обсяг приблизно 250 слів. Структура: вступ, приклад з літератури, приклад з життя та висновок.», де {0} – тема висловлювання, що призвів до створення текстів, які зображено на рисунку 4.4.

	A	B	C
1	id	topic	text
2	1	Як стереотипи впливають на життя людини?	Стереотипи – це певний вид узагальнення, який може бути корисним для певних ситуацій, проте в найбільшій частині випадків во На прикладі літературогої ілюстрації - Марк Твен написав повість "Пригоди Тома Сойєра", у якій головний герой та його друзі діал В подобной строке ми знайомимося з прикладом з реального життя. Під час промовистого звіту висвітлені цифри про кількість жі У висновку можна відзначити, що стереотипи впливають на наше життя, певна міра з якої корисна, але в той же час, вони мають
3	2	Чому сьогодні толерантність – це яскравий вияв	Сьогодні толерантність – це не просто слово, а важливий принцип, який є неодмінною умовою розвитку будь-якої сучасної демок Прикладом толерантності може слугувати література. Наприклад, роман "Мартін Іден" Джека Лондона, у якому автор зображує з Більш конкретним прикладом може бути ситуація, яка трапилася у Великій Британії. На початку 2020 року, випадкова людина від Отже, толерантність є однією з ключових умов для створення мультикультурного середовища та його збереження. Вона робить д
4	3	Які складники креативності, на Вашу думку, є ос	Креативність є невід'ємною складовою успіху в сучасному світі. Кожна людина має потенціал бути креативною, але не кожен здат По-перше, важливим складником креативності є уявність. Це здатність мислити вільно та фантазувати. Уявність дає можливість зн По-друге, самоствердження та самодисципліна є ключовими складниками креативності. Креативні люди мають сильні внутрішні По-третє, на думку, одним з основних складників креативності є вміння ризикувати. Креативні люди мають захоплююче інтелекту Отже, висновок очевидний: креативність є ключем до успіху в сучасному світі, а уявність, самовизначення та ризик є складниками
5	4	Який зі способів подолання конфліктів є найбільш	Відносини між людьми завжди були складним процесом. Вони складаються зі співлади, співпраці, але не можна уникнути й конф Такий підхід був показаний у романі "Війна і мир" Л. Толстого. У книзі описані різні сцени конфліктів, які були виникли між персон Одним з найбільш вражаючих прикладів позитивного вирішення конфлікту зазвичай є життєвий досвід. У моєму випадку, це був Отже, можна висновувати, що здатність до сприйняття іншої сторони та розуміння точки зору партнера є найбільш ефективним сп
6	5	Як можна бути патріотом, живучи поза межами (	Бути патріотом - це бути вірним своїй країні та її цінностям. Але як це можна зробити, коли живеш поза межами своєї країни, якщ У романі Вільяма Голдинга "Шалені бабуї" є персонаж на ім'я Роджер, який відчуває велику кількість гніву, і його єдина мета стає Одним з прикладів більш актуального часу може бути ситуація з іспанською громадянкою, яка знаходилася на Україні під час Рев Бути патріотом поза межами своєї країни може бути викликом, але можна стверджувати, що це можливо. Прояви патріотизму не
7	6	Що краще в житті: вірити у власні сили й ризикує	Віра в себе є однією із найважливіших складових успіху в житті. І хоча багато людей згодяться з цим твердженням, процес ризику Один зі сторінників віри в себе - письменник і лідер громадянських рухів Маріо Варгас Льоса. У своїй книзі "Брахмапутра" він гов Аналогічна ситуація мала місце в житті відомого підприємця Річарда Бренсона, який заснував "Virgin Group". У своїй книзі "Щоб д Також варто згадати вченого-стипендіата і фізика Марка Ейнштейна, який був переконаний, що важливо мати віру в свої здібност Отже, віра в свої сили та ризик - ключ до успіху. Іноді потрібно приймати на себе нові виклики та ризикувати, а не боятися нових

Рисунок 4.4 – Тексти сформовані GPT

Важливо зазначити, що створення тексту за допомогою GPT може бути недосконалим і може потребувати додаткового редагування та коригування, щоб зробити його більш зрозумілим та чітким.

4.2 Експеримент

Порівняння моделей для класифікації текстів будемо виконувати у Google Colab з використанням таких бібліотек, як Scikit-learn, Keras, nltk, numpy, pandas тощо [16], [22].

Обробка даних перед класифікацією може включати в себе видалення стоп-слів. Стоп-слова – це слова, які часто зустрічаються в тексті, але не мають суттєвого значення для розуміння змісту тексту, такі як «і», «в», «на», «з», «що» тощо.

Стоп-слов [18] української мови не має у звичайній бібліотеці «nltk». Для цього завантажимо ці слова зі стороннього ресурсу, як на рисунку 4.5.

```
url = 'https://raw.githubusercontent.com/olegdubetcky/Ukrainian-Stopwords/main/ukrainian'
r = requests.get(url)
with open(nltk.data.path[0]+'/corpora/stopwords/ukrainian', 'wb') as f:
    f.write(r.content)
# Retrieve HTTP meta-data
print(r.status_code)
print(r.headers['content-type'])
print(r.encoding)

stopwords = stop_words.words("ukrainian")
print(len(stopwords))
```

```
200
text/plain; charset=utf-8
utf-8
166
```

Рисунок 4.5 – Завантаження стоп-слів

Видалення стоп-слів може поліпшити якість класифікації, оскільки ці слова не мають інформаційної цінності. На рисунку 4.6 зображено текст та змінений текст без стоп-слів.

```
df[["text", "text_sw"]][1:2]
```

1 entry Filter

index	text	text_sw
62	У житті кожна людина зустрічається з багатьма людьми, і з часом виникає потреба у справжніх друзях, яких можна назвати своїми близькими людьми. Але як знайти таких друзів, яких можна довіряти і на яких можна розраховувати? Хто ж вони справжні друзі, а хто просто користується твоєю добротою? Про це питання думав наш великий український письменник Тарас Шевченко, який у своєму творі "Катерина" зобразив історію про дівчину, яку по-справжньому любив тільки один її друг. І це правдивий приклад того, як становиться зрозуміло, хто справжній друг. Також, щоб визначити справжнього друга, можна дивитися на те, як він поводиться зі вами, коли у вас виникають складнощі. Справжній друг буде завжди поруч і допоможе вам пережити будь-яку складність. Він не просто скаже, що все буде добре, а реально підтримає свою підтримку діями і вчинками. В кінці кінців, якщо ви знайшли справжнього друга, варто пам'ятати, що це дуже важливі люди в житті. Це люди, які завжди поруч і завжди готові допомогти. Вони можуть допомогти вам важливими рішеннями, вдихнути у ваше життя натхнення і дати силу пройти будь-який етап життя. Тому будьте вантажуватися на стосунки з друзями, думайте про те, які вони є справжніми, і не забувайте пам'ятати про те, що вони завжди поруч з вами.	У житті кожна людина зустрічається багатьма людьми , часом виникає потреба справжніх друзях , яких назвати своїми близькими людьми . Але знайти таких друзів , яких довіряти яких розраховувати ? Хто справжні друзі , користується твоєю добротою ? Про це питання думав наш великий український письменник Тарас Шевченко , своєму творі `` Катерина '' зобразив історію дівчину , яку по-справжньому любив друг . І це правдивий приклад , становиться зрозуміло , справжній друг . Також , визначити справжнього друга , дивитися , поводиться вами , виникають складнощі . Справжній друг поруч допоможе пережити будь-яку складність . Він скаже , все , реально підтримає підтримку діями вчинками . В кінці кінців , знайшли справжнього друга , варто пам'ятати , це важливі люди житті . Це люди , які поруч готові допомогти . Вони можуть допомогти важливими рішеннями , вдихнути ваше життя натхнення дати силу пройти будь-який етап життя . Тому будьте вантажуватися стосунки друзями , думайте , які справжніми , забувайте пам'ятати , поруч вами .

Рисунок 4.6 – Видалення стоп-слів

Розділення даних на тренувальні та валідаційні - це важливий крок в машинному навчанні, щоб оцінити ефективність моделі та запобігти перенавчанню. Для цього ми можемо використовувати функцію train_test_split з бібліотеки scikit-learn.

На рисунку 4.7 приклад коду для розділення даних на тренувальні та валідаційні з використанням функції train_test_split.

```
# split the dataset into training and validation datasets
# ML classic
train_x_sw, valid_x_sw, y_train_sw, y_valid_sw = model_selection.train_test_split(df[TEXT+"_sw"],
                                                                                  df[LABEL],
                                                                                  random_state=42,
                                                                                  stratify=df[LABEL],
                                                                                  test_size=0.2)

# For Embeddings
train_x, valid_x, y_train, y_valid = model_selection.train_test_split(df[TEXT],
                                                                      df[LABEL],
                                                                      random_state=42,
                                                                      stratify=df[LABEL],
                                                                      test_size=0.2)

# label encode the target variable
encoder = preprocessing.LabelEncoder()
train_y_sw = encoder.fit_transform(y_train_sw)
valid_y_sw = encoder.fit_transform(y_valid_sw)
train_y = encoder.fit_transform(y_train)
valid_y = encoder.fit_transform(y_valid)
```

Рисунок 4.7 – Розділення датасету

Ми розділили дані на тренувальні та валідаційні за допомогою функції train_test_split з параметром test_size=0.2, який означає, що 20% даних будуть призначені для валідації. Також ми вказали параметр random_state=42, щоб забезпечити відтворюваність результатів. Нарешті, ми вивели розміри тренувальних та валідаційних даних.

Count Vectorizing – це модифікація методу One-Hot Encoding, яка дозволяє розв'язати деякі з цих проблем. Замість того, щоб встановлювати значення елементу вектора на 1, якщо слово міститься в документі, ми рахуємо, скільки разів слово з'являється в документі. Значення кожного елементу вектора відповідає кількості входжень слова у документ. Якщо слово не з'являється в документі, значення елементу вектора буде дорівнювати 0.

На рисунку 4.8 наведено приклад коду для застосування Count Vectorizing з використанням бібліотеки scikit-learn.

```
%%time
# create a count vectorizer object
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(df[TEXT+"_sw"])

# transform the training and validation data using count vectorizer object
xtrain_count =  count_vect.transform(train_x_sw)
xvalid_count =  count_vect.transform(valid_x_sw)
```

```
CPU times: user 28.9 ms, sys: 4.28 ms, total: 33.1 ms
Wall time: 41 ms
```

Рисунок 4.8 – Count Vectorizing

Ми спочатку створили екземпляр CountVectorizer. Потім використали метод fit_transform для тренувальних даних train_x_sw, щоб навчити модель розуміти, які слова з'являються у документах, і побудувати векторні представлення для кожного документа на основі кількості входжень кожного слова. Результатом є матриця xtrain_count, де кожен рядок

відповідає окремому документу, а кожний стовпчик відповідає окремому слову в словнику. Тіж самі дії виконуємо і для валідаційних даних.

Створимо TF-IDF [15] вектори трьох видів, що зображено на рисунку 4.9:

– word level TF-IDF – використовується для створення вектора документа, що відображає частоту кожного слова у документі;

– n-gram level TF-IDF – використовується для створення вектора документа, який враховує частоту кожної n-грами у документі;

– characters n-gram level TF-IDF – використовується для створення вектора документа, який враховує частоту кожної n-грами символів у документі.

```
%%time
# word level tf-idf
tfidf_vect = TfidfVectorizer(analyzer='word',
                             token_pattern=r'\w{1,}',
                             max_features=10000)
tfidf_vect.fit(df[TEXT])
xtrain_tfidf =  tfidf_vect.transform(train_x_sw)
xvalid_tfidf =  tfidf_vect.transform(valid_x_sw)
print("word level tf-idf done")
# ngram level tf-idf
tfidf_vect_ngram = TfidfVectorizer(analyzer='word',
                                   token_pattern=r'\w{1,}',
                                   ngram_range=(2,3),
                                   max_features=10000)
tfidf_vect_ngram.fit(df[TEXT])
xtrain_tfidf_ngram =  tfidf_vect_ngram.transform(train_x_sw)
xvalid_tfidf_ngram =  tfidf_vect_ngram.transform(valid_x_sw)
print("ngram level tf-idf done")
# characters level tf-idf
tfidf_vect_ngram_chars = TfidfVectorizer(analyzer='char',
                                         ngram_range=(2,3),
                                         max_features=10000)
tfidf_vect_ngram_chars.fit(df[TEXT])
xtrain_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(train_x_sw)
xvalid_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(valid_x_sw)
print("characters level tf-idf done")
```

```
word level tf-idf done
ngram level tf-idf done
characters level tf-idf done
CPU times: user 343 ms, sys: 5.19 ms, total: 348 ms
Wall time: 356 ms
```

Рисунок 4.9 – Три види TF-IDF

FastText – це бібліотека для векторизації тексту та класифікації тексту, яка розроблена Facebook AI Research (FAIR). Вона базується на підході нейронної мережі згортки та нейронної мережі з керованим зворотним поширенням, що дозволяє працювати зі словами, які не зустрічаються в

обучальному корпусі. FastText можна використовувати для векторизації тексту на рівні слів, n-грам та символів [15].

Щоб використовувати FastText, спочатку необхідно завантажити бібліотеку та модель. Можна завантажити модель для різних мов. Завантажимо модель для української. створимо вектори за допомогою бібліотеки FastText, як зображено на рисунку 4.10.

```
%%time
maxlen=300
# create a tokenizer
token = Tokenizer()
token.fit_on_texts(df[TEXT])
word_index = token.word_index

# convert text to sequence of tokens and pad them to ensure equal length vectors
train_seq_x = tf.keras.utils.pad_sequences(token.texts_to_sequences(train_x), maxlen=maxlen)
valid_seq_x = tf.keras.utils.pad_sequences(token.texts_to_sequences(valid_x), maxlen=maxlen)

# create token-embedding mapping
embedding_matrix = np.zeros((len(word_index) + 1, maxlen))
words = []
for word, i in tqdm(word_index.items()):
    embedding_vector = pretrained.get_word_vector(word) #embeddings_index.get(word)
    words.append(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

```

```
100%|██████████| 5506/5506 [00:00<00:00, 69531.90it/s]CPU times: user 121 ms, sys: 11.3 ms, total: 132 ms
Wall time: 132 ms
```

Рисунок 4.10 – Використання FastText

Далі використатуємо методи векторизації на моделі машинного навчання та FastText для деяких архітектур штучних нейронних мереж.

Метод векторизації – це процес перетворення тексту на числові вектори, які можуть бути використані для тренування моделей машинного навчання.

FastText є бібліотекою для роботи з текстовими даними, яка використовує метод Word Embeddings. Вона може вирішувати проблему з різними формами слова, так як вона розбиває слова на підрядки (n-грами) та створює вектори для цих підрядків. Це дозволяє моделі розуміти семантичні зв'язки між словами та уникнути проблем зі словами, які не були побачені в тренувальному наборі даних.

Щодо архітектур штучних нейронних мереж, такі як Convolutional Neural Networks (CNN) та Recurrent Neural Networks (RNN), вони можуть використовуватись для класифікації тексту. CNN використовуються для аналізу локальних особливостей тексту, тоді як RNN можуть моделювати контекст залежностей між словами.

Таким чином, поєднуючи метод векторизації, FastText та архітектури штучних нейронних мереж або алгоритми машинного навчання [19], [20], можна досягти покращення результатів класифікації текстів.

За результатами інших досліджень [10], [24], [27] відомо, що трансформери є найбільш ефективним методом для класифікації текстів, зокрема для виявлення синтетичних текстів. Проте, для навчання таких моделей необхідно значна кількість даних, що відноситься і до глибинних мереж. Зважаючи на те, що в нашому випадку ми маємо невеликий датасет, дослідження показало, що зображено на таблиці 4.1, що необхідно розглянути інші методи машинного навчання, які можуть бути ефективними для класифікації текстів на основі невеликої кількості даних.

Таблиця 4.1 – Точність деяких найефективніших моделей

Модель	**Точність**
XGB_Count_Vectors	0.9921
XGB_WordLevel_TF-IDF	0.9846
SGD_Count_Vectors	0.9818
SVM_Count_Vectors	0.9636
LR_Count_Vectors	0.9454
LR_WordLevel_TF-IDF	0.9451
XGB_CharLevel_TF-IDF	0.9444

З таким датасетом можливо лише використання методів векторизації текстів, таких як TF-IDF та Word2Vec, для підготовки даних до навчання

моделей машинного навчання, таких як SVM, XGBoost, SGD. Ці методи мають достатню ефективність для роботи з невеликими датасетами і можуть допомогти у виявленні синтетичних текстів.

Таким чином, зважаючи на обмежену кількість даних, ці методи можуть забезпечити найкращу точність класифікації. Використання глибинного навчання, зокрема трансформерів та глибоких нейронних мереж, може бути неефективним на малих датасетах через нестачу даних для навчання таких моделей. Таким чином, використання методів векторизації текстів з моделями машинного навчання, зокрема як SVM, XGBoost, SGD, може бути оптимальним підходом для виявлення синтетичних текстів на малих датасетах. Однак, вирішення задачі виявлення синтетичних текстів залишається складним завданням, що потребує подальших досліджень та розробок методів.

ВИСНОВКИ

Отже, синтетичні тексти стають дедалі важливішим ресурсом для дослідження класифікації текстів, надаючи цінні дані для навчання та тестування алгоритмів машинного навчання. Однак, хоча моделі глибокого навчання показали багатообіцяючі результати на великих наборах даних, вони не підходять для невеликих наборів даних через свою високу складність і потребу в великих навчальних даних. Натомість алгоритми машинного навчання, такі як SVM, XGBoost і SGD, виявилися найефективнішими для класифікації тексту на невеликих наборах даних.

Результати дослідження, яке показало, що XGBoost разом з Word Level TF-IDF та Count Vectorizing є найефективнішими методами для класифікації текстів, є дуже цінними для обробки природньої мови. Метод Word Level TF-IDF дозволяє векторизувати текст за допомогою вагованого підходу, що дає змогу більш точно оцінити важливість окремих слів в тексті. Count Vectorizing же використовує просту підрахунок кількості входжень слів у текст, що може бути ефективним для більш простих задач класифікації. XGBoost, з іншого боку, є потужним алгоритмом машинного навчання, який може працювати з різними типами даних та показує дуже хороші результати на багатьох задачах класифікації.

Оскільки дослідження в галузі класифікації текстів тривають, важливо враховувати сильні та слабкі сторони різних моделей і підходів, щоб вибрати найбільш підходящі методи для кожної конкретної задачі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.).

2. Генерація текстів: перевіряємо прогрес AI-моделі від GPT до ChatGPT. URL: https://dou.ua/forums/topic/41509/ (дата звернення 15.04.2023).

3. Sandra K., Shubham S. GPT-3: Building Innovative NLP Products using LLMs.

4. Manning C. D., Schütze H. Foundations of Statistical Natural Language Processing.

5. Goldberg Y. Neural Network Methods for Natural Language Processing.

6. Pang B., Lee L. . Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval.

7. Collobert R., Weston, J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.

8. Кублик С., Сабу Ш. GPT-3. Посібник з використання API Open AI

9. Weixin L., Mert Y., Yining M., Eric W., James Z. GPT detectors are biased against non-native English writers.

10. Jonas B., Jan P., Terry R., Bela G. Paraphrase Detection: Human vs. Machine Content.

11. Eric M., Yoonho L., Alexander K., Christopher D. M., Chelsea F. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature

12. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. 2013.

13. Piotr B., Edouard G., Armand J., Tomas M. Enriching word vectors with subword information.

14. Russell S., Norvig P. Artificial Intelligence: A Modern Approach

15. Michie D., Spiegelhalter D.J., Taylor C.C. Machine Learning, Neural and Statistical Classification.

16. Bird S., Klein E., Loper E. Natural Language Processing with Python.

17. Sebastiani F. Machine Learning in Automated Text Categorization.

18. Kamran K., Kiana Jafari M., Mojtaba H., Sanjana M., Laura B., Donald B. Text Classification Algorithms: A Survey.

19. Socher R., Perelygin A., Wu J., Chuang J., Manning C. D. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.

20. Abu-Mostafa Yaser S., Magdon-Ismail M., Hsuan-Tien L. Learning From Data

21. Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features.

22. Chollet F. Deep Learning with Python.

23. Kim Y. . Convolutional Neural Networks for Sentence Classification.

24. Johnson R., Zhang, T. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding.

25. Lapan M. Deep Reinforcement Learning Hands-On

26. Lai S., Xu L., Liu K., Zhao J. Recurrent Convolutional Neural Networks for Text Classification

27. Andres G.-S., Cristian B., Jose M. G.-P. Understanding Transformers for Bot Detection in Twitter.

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	54 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Безродний В.В.			Дослідження методів виявлення синтетичних текстів
Перевірив		Турута О.П.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-21-1	122
ХНУРЕ кафедра ШІ	