МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Інформаційні системи та технології
Освітня програма	Інформаційні системи та технології
Група	6.04.126.010.18.1

ДИПЛОМНИЙ ПРОЕКТ

на тему: «Розроблення модулів інформаційної системи класифікації текстів»

Виконала: студентка Катерина ПОТАПОВА

Керівник: к.т.н. Олена ПЕРЕДРІЙ

Консультант: –

Рецензент: к.т.н, доц. кафедри Інформатики
Харківського національного університету радіоелектроніки
доц. Валентин ЛЮБЧЕНКО

Харків – 2022 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет Інформаційних технологій
Кафедра Інформатики та комп'ютерної техніки
Освітній ступінь Бакалавр
Спеціальність 126 "Інформаційні системи та технології"

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформатики та комп'ютерної техніки
_____ проф. Сергій УДОВЕНКО
" " 2022 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ
Потапової Катерини

1. Тема проекту: «Розроблення модулів інформаційної системи класифікації текстів» та керівник проекту: Передрій Олена, кандидат технічних наук затверджені наказом ректора від "01" лютого 2022 р. № 178-С.

2. Строк подання студентом проекту: "09" червня 2022 р.

3. Вихідні дані до проекту: методи обробки неструктурованого тексту, методи класифікації текстів, вхідні бази текстових даних, бібліотеки Keras, TensorFlow, NumPy, літературні джерела.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області

Розділ 2. Розробка алгоритму класифікації текстових даних

Розділ 3. Реалізація алгоритму класифікації текстових даних

5. Перелік графічного матеріалу: актуальність проблеми класифікації тексту,

методи попередньої обробки тексту**,** етапи класифікації текстових даних, етапи програмної реалізації алгоритму, аналіз експериментальних результатів.

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ім'я та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання: "01" лютого 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1	Розроблення плану дипломного проекту, ознайомлення з літературними джерелами	01.02.2022	
2	Аналіз предметної області	01.03.2022	
3	Ознайомлення з методами попередньої обробки тексту	10.03.2022–31.03.2022	
4	Розроблення програмних модулів, проведення експериментів	01.04.2022–24.04.2022	
5	Попередня перевірка дипломного проекту керівником	25.04.2022–30.04.2022	
6	Оформлення пояснювальної записки	01.05.2022–20.05.2022	
7	Перевірка дипломного проекту на плагіат	05.06.2022	
8	Підготовка презентації та доповіді	07.06.2022	
9	Подання Голові Екзаменаційної комісії щодо захисту дипломного проекту	09.06.2022	

Студент Катерина ПОТАПОВА

Керівник проекту Олена ПЕРЕДРІЙ

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 64 сторінок, 42 рисунки, 3 таблиці, 2 додатки, 26 джерел.

Об'єктом дослідження є методи класифікації неструктурованих текстових даних.

Метою роботи є дослідження методів та підходів до класифікації текстових даних, створення нейронної мережі для класифікації текстів та оцінка її роботи.

Застосовано методи комп'ютерного моделювання, аналізу неструктурованих текстових даних.

У результаті виконання роботи здійснено програмну реалізацію методу бінарної класифікація з використанням нейронних мереж, збирання даних з сайтів новин та їх класифікація за тематичними групами.

Результати роботи було апробовано у вигляді тез доповіді на конференції, а також вони можуть бути використані як частина автоматизованої системи класифікації великого обсягу текстових даних за категоріями.

ІНФОРМАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, ЗАДАЧА КЛАСИФІКАЦІЇ, КЛАСИФІКАЦІЯ ТЕКСТІВ, АЛГОРИТМИ КЛАСИФІКАЦІЇ.

ABSTRACT

The bachelor's thesis report: 64 pages, 42 figures, 3 tables, 2 appendices, 26 sources.

The object of the study is the methods of classification of non-structured text data.

The purpose of the work is to study methods and approaches to the classification of text data, to create a neural network for the classification of texts and to evaluate its work.

Methods of computer modeling, analysis of non-structured text data are applied.

As a result of the work the program realization of the method of binary classification with the use of neural networks, data collection from news sites and their classification by thematic groups was carried out.

The results of the work were tested in the form of a report at the conference, they can be used as part of the automated system of classification of large volume of text data by categories.

INFORMATION SYSTEM, MACHINE LEARNING, CLASSIFICATION TASK, TEXT CLASSIFICATION, CLASSIFICATION ALGORITHMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CNN – Згорткові нейронні мережі.

EDA – Експлораторний аналіз даних (Exploratory Data Analysis).

LSTM – Long Short-Term Memory.

MLP – Нейронні мережі з прямою передачею.

NB – Наївний байесовий класифікатор.

NLP – Обробка природної мови.

NLTK – Natural Language Toolkit.

ReLU – Rectified Linear Unit.

RNN – Рекурентні нейронні мережі.

SVM – Метод опорних векторів (англ. Support Vector Machine).

ВСТУП

Сучасний світ потопає у великих обсягах інформації, об’єм якої стрімко збільшується. Перехід від паперових носіїв інформації до цифрових, викликаний використанням інформаційних технологій, відкриває перспективи для роботи з обсягом інформації, що постійно зростає, і можливістю вилучення знань із масиву даних, який слабо структурований.

Незважаючи на відносно невеликий термін активного застосування в промисловості цифрових носіїв, обсяг даних зростає кожного року у геометричній прогресії. Це є наслідком перетворення та перекладу даних із різних областей життєдіяльності людини у цифровий вигляд.

По оцінкам, 80% інформації є неструктурованою, при чому текст є одним з найбільш розповсюджених типів неструктурованих даних. Через характер текстових даних їх аналіз, розуміння, організація та сортування є важким та трудомістким завданням, тому більшість компаній не використовують текстові данні у повному розмірі [1].

На сьогоднішній день людині все складніше стає аналізувати та класифікувати дані за категоріями. Класифікація текстів займає багато часу, адже для цього треба не тільки прочитати текст, а й проаналізувати його та обрати відповідну категорію із множини доступних. Коли мова йде про велику кількість даних, то часто виникають проблеми, не вистачає людських ресурсів для їх обробки.

Зростання інформації і одночасне збільшення доступних потужних комп’ютерів дозволяє застосовувати сучасні методи для вирішення задачі класифікації. У даному дипломному проекті пропонується автоматизувати процес класифікації текстів.

Класифікація тесту за допомогою машинного навчання вирішує ці проблеми. Використовуючи текстові класифікатори, компанії можуть автоматично структурувати усі види релевантного тексту, починаючи з електронних листів, юридичних документів до соціальних мереж, чат-ботів тощо

швидким та економічним способом. Це дозволить компаніям економити час на аналізі текстових даних, автоматизувати бізнес-процеси та приймати бізнес-рішення на основі даних. Загалом, класифікація текстів відіграє важливу роль в отриманні та підсумовуванні інформації, пошуку тексту та відповідей на запитання [1].

Дипломний проект присвячений аналізу існуючих систем класифікації текстових даних, виявленню недоліків та розробці модулів інформаційної системи для бінарної класифікація відгуків на кінофільми та класифікації новин за категоріями.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Змістовий опис і аналіз предметної області

Класифікація тексту – це метод машинного навчання, що полягає у віднесенні документа до однієї з декількох категорій на підставі змісту документа. Текстові класифікатори можна використовувати для організації, структурування та категоризації будь-якого тексту [1].

Наприклад, нові статті можуть бути організовані за темами, чати організовані за мовами, згадування бренду організовано за настроєм і т. д.

Класифікацію тексту можна виконувати двома способами: ручним або автоматичним.

Класифікація тексту вручну включає в себе людину, яка інтерпретує зміст тексту і класифікує його відповідно. Цей метод може дати гарний результат, але він трудомісткий та дорогий.

Автоматична класифікація тексту використовує машинне навчання, обробку природної мови (NLP) та інші методи, засновані на штучному інтелекті, щоб автоматично класифікувати текст більш швидко, економічно й точно.

Текст може бути багатим джерелом інформації, але знаходження важливої може бути важливим та трудомістким завданням через його неструктуровану природу [1].

У дипломній роботі пропонується автоматизоване рішення задачі класифікації текстів.

Завдяки досягненням у галузі обробки природної мови та машинного навчання, які підпадають під великий діапазон штучного інтелекту, сортування текстових даних стає простішим.

За допомогою автоматичного аналізу структури тексту, компанії можуть автоматизувати процеси та знаходити корисні відомості, що ведуть до більш ефективного прийняття рішень.

1.2 Нейронні мережі

Нейронні мережі відображають поведінку людського мозку, дозволяючи комп'ютерним програмам розпізнавати патерни і вирішувати спільні проблеми у штучному інтелекті, машинному навчанні і глибокому навчанні (Deep Learning).

Нейронні мережі ідеально підходять для того, щоб допомогти людям вирішити складні проблеми в реальних ситуаціях. Вони можуть вивчати і диференціювати нелінійні і складні зв'язки між входами і виходами; узагальнювати і робити висновки; виявляти приховані взаємозв'язки, шаблони і передбачення; моделювати сильно змінні дані (наприклад, фінансові часові ряди) і варіації, необхідні для передбачення рідкісних подій (наприклад, виявлення шахрайства).

1.2.1 Типи нейронних мереж

Нейронні мережі можна розділити на різні типи [2], які використовуються для різних цілей. Принаймні це не повний список типів, нижче наведено найпоширеніші типи нейронних мереж, з якими можливо зіткнутися при використанні.

Ці три класи мереж забезпечують велику гнучкість і протягом десятиліть вони довели свою корисність і надійність в широкому діапазоні завдань. Вони також мають багато підтипів, що дає можливість адаптувати їх для вирішення різних задач прогнозування на різних наборах даних.

Нейронні мережі з прямою передачею, або багатошарові перцептрони (MLP). Нейронні мережі з прямою передачею складаються з вхідного шару, прихованого шару або шарів, і вихідного шару. Ці нейронні мережі часто називають MLP, хоча важливо зазначити, що вони насправді складаються з сигмоїдальних нейронів, а не перцептронів, оскільки більшість проблем реального світу не є лінійними. Дані, як правило, представлені в цих моделях для

вивчення, і вони є основою для комп'ютерного зору, обробки природної мови та інших нейронних мереж.

Згорткові нейронні мережі (CNN) схожі на нейронні мережі прямого поширення, але вони зазвичай використовуються для розпізнавання об’єктів, розпізнавання образів та/або комп'ютерного зору. Ці мережі використовують принципи лінійної алгебри, в тому числі множення матриць, для ідентифікації патернів на зображенні.

Рекурентні нейронні мережі (RNN) визначаються наявністю первинного зв'язку. Ці алгоритми навчання в основному використовуються для використання даних часових рядів при прогнозуванні майбутніх результатів, таких як прогнози фондового ринку або прогнози продажів.

Узагальнено деякі відмінності між різними типами нейронних мереж представлені в таблиці 1.1 [3]:

Таблиця 1.1 – Порівняння нейронних мереж

	MLP	RNN	CNN
Дані	Табличні дані	Дані послідовності	Дані зображення
Рекурентні з'єднання	Ні	Так	Ні
Розподіл параметрів	Ні	Так	Так
Розподіл параметрів	Ні	Ні	Так
Зникаючий та вибуховий градієнт	Так	Так	Так

1.2.2 Як працюють нейронні мережі

Вузол моделюється за допомогою нейрона в мозку людини. Як і нейрони, вузли активуються з достатньою кількістю стимулів або вхідних даних. Ця активація поширюється по мережі, створюючи відповідь на стимули (виходи).

Зв'язки між цими штучними нейронами діють як прості синапси, що дозволяють передавати сигнали від одного до іншого. Сигнали перетинають шари, проходячи від першого входу до останнього вихідного шару, і обробляються вздовж шляху.

Коли перед нейронами ставиться завдання чи проблема, яку необхідно вирішити, вони проводять математичні розрахунки, щоб з'ясувати, чи достатньо інформації для передачі наступному нейрону. Простіше кажучи, вони зчитують усі дані та з'ясовують, де існують найсильніші взаємозв'язки. У найпростішому типі мережі отримані дані підсумовуються, і якщо сума перевищує певне граничне значення, нейрон "вистрілює" та активує нейрони, до яких він підключений.

У міру збільшення кількості прихованих шарів у нейронній мережі формуються глибокі нейронні мережі. Архітектури глибокого навчання виводять прості нейронні мережі на новий рівень. Використовуючи ці шари, фахівці з дослідження даних можуть створювати власні мережі глибокого навчання, які забезпечують машинне навчання, що дозволяє навчити комп'ютер точно імітувати людські завдання, такі як розпізнавання мови, ідентифікація зображень або складання прогнозів. Не менш важливим є те, що комп'ютер може навчатися самостійно, розпізнаючи закономірності в багатьох шарах обробки [4].

Дані надходять у нейронну мережу через вхідний шар, який передає їх на приховані шари. Обробка відбувається у прихованих шарах через систему залежних зв'язків. Вузли в прихованому шарі поєднують дані із вхідного шару з набором коефіцієнтів і привласнюють відповідні ваги входам. Величина ваги допомагає нам надати більше значення одному зв’язку над іншими. Потім ці добутки коефіцієнтів і ваг підсумовуються. Сума проходить через функцію активації вузла, яка визначає ступінь, в якій сигнал повинен пройти далі через мережу, щоб вплинути на кінцевий вихід. Тобто нейрон надішле сигнал наступному шару, якщо результат перевищить певне число. Нарешті, приховані шари з'єднуються з вихідним шаром, звідки виходять вихідні дані.

1.3 Проблема класифікації тексту.

Розвиток та широке використання інформаційних технологій має великий вплив на усі області сучасного життя, включаючи економіку, управління, науку та освіту. Сьогодні обсяг інформації, що зберігається у традиційному вигляді, ускладнює ефективну роботу з нею — зберігання, пошук, облік тощо. Вирішення проблем ефективного аналізу та обробки інформації полягає у способі використання сучасних комп'ютерних та інформаційних технологій і потребує подання інформації в електронному вигляді. Електронне представлення інформації дає можливість зберігати інформацію найбільш надійним і компактним способом, поширювати її набагато швидше і ширше.

Розповсюдженими прикладами електронних репозитаріїв для текстової інформації є електронні архіви, бібліотеки, системи управління документами та інші. Обробка та аналіз текстового контенту зазвичай виконувався вручну, але зі збільшенням обсягу інформації вони стали неефективними і виникала необхідність автоматизації цих процесів. Однією з актуальних задач аналізу текстових даних є класифікація – визначення тематики заданого тексту і віднесення його до однієї з наперед визначених категорій.

Класифікація текстових даних – це визначення належності текстових даних будь-якій темі, якій присвячений текст. Класифікація по темах часто використовується для фільтрації текстових даних, коли необхідно відсікти записи, що відносяться до тем, які не становлять інтересу для аналізу.

Вирішення завдань автоматичної класифікації текстів в останні роки стало одним із пріоритетних напрямків розвитку досліджень в області інформаційного пошуку та штучного інтелекту. Засоби автоматичної класифікації текстів знаходять застосування не тільки при відборі найбільш релевантних результатів пошукових запитів, але і при вирішенні таких прикладних завдань, як фільтрація спаму, складання персональних добірок новин, автоматичне анотування, зняття неоднозначності при автоматичному перекладі, визначення мови тексту.

1.4 Типи класифікації.

Класифікаційні завдання – це способи розв'язання різних прикладних задач. На практиці часто використовуємо алгоритми машинного навчання, щоб звести реальні задачі до завдань класифікації для їх вирішення. Прикладом такої задачі є ідентифікація спаму. Щоб вирішити цю проблему, ми часто класифікуємо електронні листи як "спам" або "Не спам". Отже потрібно аналізувати типи задач класифікації з конкретної предметної області, і кожен тип задач може використовувати спеціальний метод моделювання. Це впливає на остаточне вирішення проблеми. Зіткнувшись із різними класифікаційними мітками зразків, особливо важливим стає визначення завдання класифікації цього типу питань.

Завдання класифікації приблизно поділяються на чотири різні типи: бінарна класифікація, завдання множинної класифікації, задачі класифікації з декількома мітками та задачі класифікації незбалансованої вибірки [5].

1.4.1 Бінарна класифікація

Бінарна класифікація – це один із типів завдань класифікації у машинному навчанні, коли ми повинні класифікувати два взаємовиключні класи.

Як правило, завдання подвійної класифікації включає один клас, що належить до нормального стану, та інший клас, що належить до ненормального стану. Наприклад, «не спам» – це нормальний стан, а «спам» – ненормальний стан. Інший приклад: «новини справжні» – це нормальний стан завдань, а «новини фальшиві» – ненормальний стан. Категорія в нормальному стані надається мітка категорії 0, а категорія в ненормальному стані призначається мітка категорії 1 [5].

Алгоритми, які зазвичай використовуються для вирішення задач бінарної класифікації, включають: k найближчих сусідів, дерево рішень, метод опорних векторів, наївний баєсів класифікатор тощо.

1.4.2 Мультикласова класифікація

Мультикласова класифікація – це завдання класифікації з більш ніж двома класами. Кожен зразок можна помітити лише як один клас.

Наприклад, класифікація з використанням ознак, витягнутих із набору зображень овочів, де кожне зображення може бути помідором, огірком або картоплею. Кожне зображення є одним зразком і позначений як один із 3 можливих класів. Мультикласова класифікація передбачає, що кожному зразку надається одна і тільки одна мітка — наприклад, один зразок не може бути одночасно помідором та огірком.

1.4.3 Багаторівнева класифікація

Класифікація з декількома мітками належить до завдань класифікації, які мають дві або більше мітки класу, де одна або кілька міток класу можуть бути передбачені для кожного прикладу.

Розглянемо приклад класифікації фотографій, де на фотографії може бути кілька об'єктів на сцені, а модель може передбачити наявність на фотографії кількох відомих об'єктів, таких як велосипед, яблуко, людина і т. д. Це відрізняється від бінарної класифікації та багатокласової класифікації, де для кожного прикладу передбачається одна мітка класу.

Зазвичай завдання класифікації з кількома мітками моделюють за допомогою моделі, яка передбачає кілька вихідних даних, причому кожен вихідний результат прогнозується як розподіл ймовірностей Бернуллі. По суті це модель, яка робить кілька прогнозів бінарної класифікації для кожного прикладу.

1.4.4 Незбалансована класифікація

Незбалансована класифікація належить до завдань класифікації, у яких кількість прикладів у кожній категорії розподіляється нерівномірно. Це часто ілюструється завданням бінарної класифікації, де більшість прикладів відносяться до класу 0 і лише кілька прикладів відносяться до класу 1. Розподіл може варіюватися за ступенем серйозності від 1:2, 1:10, 1:100 або навіть 1:1000.

1.5 Огляд та аналіз аналогів

Розглянемо найпопулярніші програми, веб-дадатки та бібліотеки, які дозволяють виконувати аналіз та класифікацію текстових даних. Було виділено наступні аналоги [6].

Apache OpenNLP — Java-бібліотека з відкритим кодом, яка використовується для обробки текстів природною мовою. OpenNLP надає такі можливості, як токенізація, сегментація речення, маркування мовлення, добування іменованого об'єкта, OpenNLP також включає максимальну ентропію і машинне навчання на основі перцептрону.

uClassify Веб-додатаки можуть бути використані для створення спам-фільтра, категоризації веб-сторінок, автоматизації підтримки електронної пошти, визначення мов, категоризації повідомлень блогів, тощо.

Carrot2. Є механізмом кластеризації результатів пошуку з відкритим кодом. Він може автоматично кластерувати невеликі колекції документів, наприклад, результати пошуку або реферати документів, в тематичні категорії. Carrot² написана на Java і розповсюджується під ліцензією BSD.

LibShortText. Програма з відкритим кодом для класифікації та аналізу короткого тексту. LibshortText може обробляти класифікацію імен, питань, речень і коротких повідомлень. Він ефективніший, ніж пакети для загального текстового аналізу. На типовому комп'ютері, обробка і вивчення 10 мільйонів коротких текстів займає всього близько півгодини. Він включає в себе інтерактивний інструмент для аналізу помилок. Виходячи з

властивості, що кожен короткий текст містить декілька слів, LibshortText надає детальну інформацію для передбачення кожного тексту.

Text2data. Програма отримує важливу інформацію з текстових документів. Створює докладні та гнучкі звіти про неструктуровані дані.

TextRazor. Програмне забезпечення TextRazor призначене для класифікації конкретної інформації з текстових баз даних. Здатне створювати нові класифікації та класифікувати існуючі на їх основі набори даних. TextRazor може класифікувати інформацію з власних текстових баз даних клієнтів. Це досягається за рахунок попереднього обстеження загальнодоступних баз знань, таких як DBpedia (похідна від Вікіпедії), Freebase і Wikidata. Програмне забезпечення також може видобувати певні види інформації на основі запитаного виду інформації за допомогою нетипового класифікатора.

У процесі аналізу було виявлено, що більшість аналогів працюють з латиницею. Для текстів написаних англійською мовою питання класифікації текстових даних дуже гарно досліджено.

1.6 Постановка задачі

Проаналізувавши готові програмні рішення можна прийти до висновку, що більшість з них націлені на роботу з англомовними текстами, а також деякі системи здатні обробляти тексти, написані російською мовою. Аналогів для роботи з українськими текстами немає.

Дана дипломна робота може бути використана для створення модулів інформаційної системи для класифікації україномовних текстів.

Для досягнення поставлених цілей треба виконати наступні завдання:

– підготувати набор даних для тестування модулю;

– дослідити та вибрати моделі нейронної мережі для автоматичної класифікації тексту;

– побудувати нейронну мережу та навчити її з використанням отриманих даних;

– провести тестування розробленого модулю;

– оцінити проведену класифікацію.

РОЗДІЛ 2. РОЗРОБКА АЛГОРИТМУ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ДАНИХ

2.1 Етапи класифікація текстових даних

Розв’язок задачі класифікації неструктурованих текстових масивів даних складається з наступних п’яти послідовних етапів [7]:

– збір даних;
– попередня обробка даних;
– дослідження та візуалізація даних;
– побудова моделі;
– оцінка моделі.

На рис. 2.1 представлено загальну схему процесу класифікації. Розглянемо кожен із його етапів.

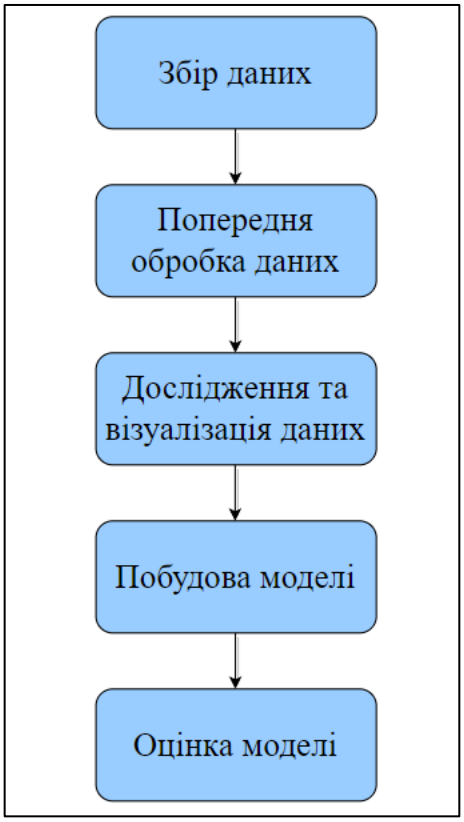


Рисунок 2.1 – Етапи класифікації текстових даних

2.1.1 Збір даних

Ми, як суспільство, генеруємо дані безпрецедентними темпами. Ці дані можуть бути числовими (температура, сума кредиту, коефіцієнт утримання клієнта), категорійними (стать, колір шкіри, вища освіта) або навіть звичайний текст (пам'ятай записки лікаря або соціологічні опитування). Збір даних — це процес збору та вимірювання інформації з різноманітних джерел. Для того, щоб використовувати зібрані дані для розробки практичного штучного інтелекту (ШІ) і рішень машинного навчання, їх потрібно зібрати і зберегти таким чином, щоб це мало сенс для конкретного бізнес завдання [8].

Збір даних дозволяє зафіксувати запис минулих подій, щоб за допомогою аналізу даних знайти повторювані закономірності. На основі цих закономірностей будуються прогностичні моделі з використанням алгоритмів машинного навчання, які шукають тенденції та передбачають майбутні зміни.

Прогностичні моделі хороші лише настільки, наскільки хороші дані, на яких вони побудовані, тому правильна практика збору даних має вирішальне значення розробки високоефективних моделей. Дані повинні бути безпомилковими та містити інформацію, необхідну для вирішення поставленого завдання.

2.1.2 Попередня обробка даних

Попередня обробка даних – це етап у процесі добування та аналізу даних, де необроблені дані перетворюються у формат, який можна зрозуміти та проаналізувати комп'ютерами та машинним навчанням.

Необроблені дані реального світу у вигляді тексту, зображень, відео тощо. Вони не тільки можуть містити помилки і невідповідності, але й часто бути неповними і не мали регулярного, рівномірного компонування.

Машини вміють обробляти та розуміти інформацію в цифровому вигляді – вони читають дані як 1 і 0. Тому обчислити структуровані дані, такі як, наприклад, цілі числа і відсотки є дуже простим завданням. Однак,

неструктуровані дані у вигляді тексту і зображень повинні спочатку бути оброблені та відформатовані перед аналізом.

Розглянемо кроки, які необхідно зробити для того, щоб дані успішно пройшли попередню обробку [9].

Оцінка якості даних (Data quality assessment)

Потрібно уважно вивчити дані і отримати уявлення про їх загальну якість, актуальність і послідовність. Практично в будь-якому наборі даних можна знайти ряд аномалій і притаманних проблем, наприклад:

Невідповідність типів даних: Коли ви збираєте дані з різних джерел, вони можуть надходити до вас у різних форматах. Хоча кінцевою метою цього процесу є переформатування даних для використання в машинах, вам все одно необхідно почати з однакових за форматом даних. Наприклад, якщо частина вашого аналізу включає сімейний дохід із кількох країн, вам доведеться перетворити кожну суму доходу на єдину валюту.

Викиди даних: Викиди можуть вплинути на результати аналізу даних. Наприклад, якщо ви усереднюєте результати тестів для класу, а один учень не відповів на жодне запитання, його 0% можуть сильно спотворити результати.

Відсутні дані: Необхідно перевірити, чи немає пропущених полів даних, порожніх місць у тексті або запитань без відповідей. Це може бути викликано людською помилкою або неповнотою даних. Щоб усунути відсутні дані, необхідно очистити дані.

Очищення даних (Data cleaning)

Очищення даних — це процес додавання відсутніх даних і виправлення, редагування або видалення недійсних або неактуальних даних з набору. Очищення даних є найважливішим кроком у процесі попередньої обробки, оскільки забезпечує готовність даних до використання для наступних потреб.

Очищення даних виправлятиме будь-які недоречні дані, які ви знайдете при оцінці якості даних. Залежно від типу даних, з якими ви працюєте, існує ряд можливих методів очищення, через які потрібно буде пропустити ваші дані.

Перетворення даних (Data transformation)

Перетворення даних це приведення даних в належний формат, зручний для їх аналізу. Зазвичай це такі етапи, як:

Агрегація. Об'єднання даних в єдиний формат.

Нормалізація. Нормалізація змінює масштаб даних на впорядкований діапазон, щоб їх можна було точніше порівняти. Наприклад, якщо ви порівнюєте втрати або збільшення співробітників в декількох компаніях (у деяких є тільки десяток працівників та інших – понад 200), то вам потрібно масштабувати їх в певному діапазоні, наприклад від -1,0 до 1.0 або від 0.0 до 1.0.

Вибір ознак. Вибір ознак – це процес визначення, які змінні (ознаки, характеристики, категорії тощо) є найважливішими для аналізу. Ці характеристики будуть використані для підготовки моделей машинного навчання. Важливо пам'ятати, що чим більше функцій ми обираємо, тим довше процес навчання, а іноді і менш точні результати, тому що деякі характеристики можуть перекривати один одного.

Дискредитація. Дискредитація об'єднує дані в менші інтерфейси. Вона дещо схожа на Data binning (Об'єднання даних), але зазвичай відбувається після очищення даних. Наприклад, при розрахунку середньодобової фізичної активності, замість використання точних хвилин і секунд, можна об'єднати дані в інтервалах 0-15 хвилин, 15-30 тощо.

Генерація ієрархії концепцій. Генерація ієрархії понять може додати ієрархію всередині та між ознаками, якої не було у вихідних даних.

Скорочення даних (Data reduction)

Чим більше даних, з якими ми працюємо, тим складніше їх аналізувати, навіть після очищення та перетворення. Залежно від поставленого завдання, у нас може бути більше даних, ніж потрібно. Особливо при роботі з аналізом тексту, більшість звичайної людської мови є зайвою або не має відношення до потреб дослідника. Скорочення даних не тільки полегшує та робить більш точним аналіз, а й скорочує обсяг збережених даних. Це також допоможе визначити найважливіші для аналізованого процесу характеристики.

Відбір за атрибутами. Подібно до дискредитації, відбір за атрибутами може об'єднати ваші дані в дрібніші пули. По суті, він об'єднує теги або ознаки, так що такі теги, як чоловік/жінка та професор, можуть бути об'єднані у професор-чоловік/професор-жінка.

Скорочення чисельності. Це допоможе при збереженні та передачі даних. Наприклад, ви можете використовувати регресійну модель, щоб використовувати тільки ті дані та змінні, які стосуються вашого аналізу.

Зниження розмірності. Це знову ж таки зменшує обсяг використовуваних даних, що допомагає полегшити аналіз і подальші процеси. Такі алгоритми, як k-найближчих сусідів, використовують розпізнавання образів, щоб об'єднати схожі дані і зробити їх більш керованими.

2.1.3 Дослідження та візуалізація даних

Дослідження даних, також відоме як експлораторний аналіз даних (EDA) – це процес, під час якого користувачі розглядають та розуміють свої дані за допомогою статистичних методів та методів візуалізації. Цей етап допомагає виявити закономірності та проблеми в наборі даних, а також вирішити, яку модель або алгоритм використовувати на наступних етапах [10].

Хоч іноді дослідники схильні витрачати більше часу на розробку архітектури моделі та налаштування параметрів, не слід ігнорувати важливість дослідження даних. Наприклад, ви розробили модель. Але якщо дані порушують припущення моделі або дані містять помилки, неможливо отримати бажані результати від моделі. Без дослідження даних можна витратити більшу частину часу на перевірку моделі, не помітивши проблеми в наборі даних.

2.1.4 Побудова моделі

Моделі машинного навчання є потужними інструментами, які використовуються для ефективного та результативного виконання життєво

важливих завдань та вирішення складних задач. Експонентне зростання обсягу даних у світі означає, що організації з різних галузей готові до впровадження моделей машинного навчання. Ці моделі мають широкий спектр застосувань, будь то машинне навчання у фінансах, проактивне відстеження банківських переказів за ознаками шахрайства, або машинне навчання в охороні здоров'я, забезпечення наступного покоління діагностичних інструментів.

2.1.5 Оцінка моделі

Оцінка моделі – це процес використання різних метрик оцінки для розуміння ефективності моделі машинного навчання, а також її сильних та слабких сторін. Оцінка моделі важлива для визначення ефективності моделі на початкових етапах дослідження, і навіть грає роль моніторингу моделі [11].

Великий набір даних випадково ділиться на три підмножини:

Навчальна множина – це підмножина набору даних, що використовується для побудови прогностичних моделей.

Валідаційний набір – це підмножина набору даних, що використовується для оцінки ефективності моделі, побудованої на етапі навчання. Він забезпечує тестову платформу для точного настроювання параметрів моделі та вибору найбільш ефективної моделі. Не всі алгоритми моделювання потребують валідаційного набору.

Тестова множина – це підмножина набору даних для оцінки ймовірної майбутньої продуктивності моделі. Якщо модель підходить до навчального набору набагато краще, ніж до тестового набору, ймовірно, причина в перенавчанні.

Розглянемо загальні метрики, що використовуються для оцінки моделей [12].

Істинно позитивний (True Positive – TP) – класифікатор правильно відніс об'єкт до класу, що розглядається.

Істинно негативний (True Negative – TN) – класифікатор вірно стверджує, що об'єкт не належить до класу, що розглядається.

Хибно позитивний (False Positive – FP) – класифікатор не правильно відніс об'єкт до класу, що розглядається.

Хибно негативний (False Negative – FN) – класифікатор не вірно стверджує, що об'єкт не належить до класу, що розглядається.

Ці чотири результати часто відображаються на матриці помилок (рис. 2.2). Наприклад, дано матрицю помилок для бінарного класифікаційного випадку. Ми створюємо цю матрицю після того, як робимо передбачення з тестових даних, а потім визначаємо кожне передбачення як один з чотирьох можливих результатів, описаних вище.

		Predicted classes	
Actual classes		Negative 0	Positive 1
	Negative 0	TN	FP
	Positive 1	FN	TP

Рисунок 2.2 – Матриця помилок класифікації

Accuracy визначається як відсоток правильних передбачень для тестових даних. Для того щоб обчислити треба поділити кількість правильних прогнозів на кількість повних прогнозів.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

Точність (precision) називається частка правильних відповідей моделі в межах класу – це частка об'єктів, що насправді належать до цього класу по відношенню до всіх об'єктів, які система віднесла до цього класу.

$$\text{precision} = \frac{\text{true positive}}{\text{true positives} + \text{ false positives}}$$

Повнота (recall) — це відсоток істинно позитивних класифікацій. Повнота показує, скільки об'єктів, які дійсно належать до позитивного класу, ми правильно передбачили

$$\text{recall} = \frac{\text{true positive}}{\text{true positives} + \text{ false negatives}}$$

Precision і recall не залежать, на відміну від accuracy, від співвідношення класів і тому використовуються в умовах незбалансованих вибірок. Часто у реальній практиці виникає проблема знаходження оптимального балансу між двома метриками. Зрозуміло, що чим вище точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота не можуть бути досягнуті одночасно, і треба знайти баланс [13].

Таким чином, має сенс поєднувати показники точності та повноти. Загальноприйнятий підхід до поєднання цих показників відомий як f-score.

F-міра – це спосіб поєднання точності та повноти моделі, і він визначається як середнє гармонійне значення точності та повноти моделі.

$$\text{F} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Ця формула дає однакову вагу до точності і повноти, тому F-міра впаде однаково при зменшені як точності, так і повноти. Можна обчислити F-міру,

надавши різну вагу до точності і повноти, якщо свідомо дати пріоритет одній з цих метрик при розробці алгоритму:

$$F_\beta = \frac{(1+\beta^2)\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}}$$

де β приймає значення в діапазоні 0<β<1, якщо ви хочете дати пріоритет точності, а при β>1 пріоритет надається повноті. При β=1 формула зводиться до попередньої і отримуємо збалансовану F-міру (також називають F1).

2.2 Обробка неструктурованих текстів

Попередня обробка тексту — це метод очищення текстових даних і підготовки його до подання моделі. Текстові дані містять шум у різних формах, таких як емоції, пунктуація, текст в іншому випадку. Коли ми говоримо про людську мову, є різні способи сказати одне й те ж, і це лише основна проблема, з якою ми маємо справу, тому що машини не розуміють слів, вони хочуть чисел, тому нам потрібно ефективно перетворювати текст на числа. Для того щоб привести текст до певної форми, структури, необхідна попередня обробка тексту, яка складається з етапів токенізації та нормалізації.

2.2.1. Токенізація

Токенізація — це процес розбивання тексту на дрібніші шматки, що називаються токенами. Ці менші частини можуть бути реченнями, словами або підсловами.

Токенізація дозволяє машинам читати тексти. Як традиційні методи, так і методи глибокого навчання для обробки природної мови покладаються на токенизацію. Вона часто є етапом попередньої обробки в більшості програм для обробки природної мови.

Наприклад, щоб підрахувати кількість слів у тексті, текст розбивається на частини за допомогою токенізаторів. У глибокому навчанні та традиційних методах токенізація використовується для розробки ознак. Наприклад, вхідний текст перед подачею до нейромережевої архітектури BERT обробляється за допомогою токенізатора підслів WordPiece [14].

Приклад токенізації відгуку на кінофільм:

Вхід: Цікавий сюжет, чудова операторська робота, прекрасні актори!

Вихід: ‘Цікавий’ ‘сюжет’ ‘чудова’ ‘операторська’ ‘робота’ ‘прекрасні’ ‘актори’.

2.2.2. Нормалізація

Нормалізація – це процес перетворення токена в його базову форму. Коли ми нормалізуємо текст, ми намагаємося зменшити його випадковість, наближаючи його до попередньо визначеного «стандарту». Це допомагає нам зменшити кількість різної інформації, з якою має працювати комп’ютер, і це підвищує ефективність. Нормалізацію можна поділити на два основних підходи: стемінг і лематизацію. Ці методи ставлять перед собою задачу привести всі використанні в тексті словоформи до однієї, але використовують різні методи для цього [15]. Наприклад, початковою формою прикметника є форма називного відмінка чоловічого роду однини, форма називного відмінка однини є початковою формою іменника.

Приклад нормалізації: найпрекрасніша – > прекрасний; сестрою – > сестра.

Стеммінг (Stemming) – це процес зведення слів до їхньої основи або кореневої форми. Метою створення основ є скорочення споріднених слів до однієї основи, навіть якщо основа не є словом зі словника.

Стемінг не є хорошим процесом для нормалізації, оскільки іноді може призвести до безглуздих слів, яких немає в словнику. Це відбувається тому, що в основному існують дві помилки при стеммінгу:

Надмірний стеммінг: коли відрізається набагато більша частина слова, ніж потрібно, що у свою чергу призводить до того, що слова неправильно скорочуються до того самого кореневого слова. Наприклад, слова ‘university’ та ‘universe’, які скорочуються до ‘univers’.

Недостатній стеммінг: відбувається, коли два або більше слів можуть бути помилково скорочені до більш ніж одного кореневого слова, коли вони повинні бути скорочені до одного і того ж самого. Наприклад, слова "data" и "datum", які скорочуються до "dat" та "datu" відповідно (замість того самого кореня "dat").

Лематизація (Lemmatization) на відміну від стеммінгу, скорочує слова до їхньої основи, правильно скорочуючи розділені слова та гарантуючи належність кореневого слова до мови. Зазвичай вона складніша, ніж стеммінг, оскільки лематизація працює над окремим словом без знання контексту. При лематизації кореневе слово називається лемою. Лемма – це канонічна форма, словникова форма або форма цитування набору слів.

2.3 Вбудовування слів

Існує багато способів обробки тексту в машинному навчанні. В роботі ми будемо використовувати вбудовування слів (word embedding) [16].

Вбудовування слів – це метод, який використовується для представлення документів у вигляді чисельного вектору. Текстові дані завжди повинні бути перетворені чисельно, при обробці тексту метод перетворення має вирішальне значення. Речення, близькі в деякому контексті, повинні відображатися в близьких числових векторах.

Словниковий запас у текстових документах відображається у вектори дійсних чисел. Семантично схожі слова відображаються близько один до одного у векторному просторі. Існують готові до використання моделі вбудовування слів, такі як Word2Vec і GloVe. Ми будемо використовувати Keras для навчання власної моделі вбудовування слів.

Для реалізації вбудовування слів бібліотека Keras містить шар, який називається Embedding(). Шар вбудовування реалізується у вигляді класу в Keras і, як правило, використовується як перший шар в послідовній моделі для задач NLP.

Метод вбудовування слів може використовуватися для виконання трьох завдань у Keras: використовуватися для навчання вбудовування слів та збереження отриманої моделі; використовуватися для вивчення вбудовування слів на додаток до виконання завдань NLP, таких як класифікація текстів, аналіз настроїв та ін.; його можна використовувати для завантаження попередньо навчених вбудованих слів та використання їх у новій моделі.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ДАНИХ

3.1 Вибір програмного забезпечення

Google Colab – це хмарний інструмент компанії Google для виконання коду мовою Python або R та створення моделей машинного навчання. Головна перевага цього інструменту в тому, що він надає повністю готове до використання програмне середовище у веб-браузері. Це означає, що нам не потрібно встановлювати будь-яке програмне забезпечення на персональний комп'ютер, щоб мати можливість програмувати або запускати будь-які програми, які ми розробляємо [17].

Використання Google Colab є корисним ще й тому, що воно звільняє наш комп'ютер від необхідності виконувати ці програми. Тобто це дозволяє нам економити ресурси нашого комп'ютера і, можливо, виконувати коди/програми, для виконання яких комп'ютеру може не вистачити ресурсів (наприклад, потужності процесора або пам'яті).

Для використання цього сервісу ми повинні підключити його до особистого або бізнес-аккаунту Google, і до нього можна отримати доступ з будь-якого браузера (наприклад, Chrome, Explorer, Firefox або Safari тощо).

Найголовніше, що Colab не вимагає налаштування, а блокноти, які ви створюватимете, можуть одночасно редагувати члени вашої команди приблизно так само, як ви редагуєте документи в Google Docs. Найбільшою перевагою є те, що Colab підтримує більшість популярних бібліотек машинного навчання, які можна легко завантажити у ваш блокнот.

Розробник може виконувати такі дії за допомогою Google Colab:

- писати та виконувати код мовою Python;
- створювати/завантажувати/обмінюватися блокнотами;
- імпортувати/зберігати блокноти з/на Google Drive;

– імпортувати/публікувати блокноти із GitHub;
– Імпортувати зовнішні набори даних;
– Інтеграція PyTorch, TensorFlow, Keras, OpenCV;
– Безкоштовний хмарний сервіс із безкоштовним GPU.

Python – мова програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python є мовою загального призначення, тобто вона може бути використана для створення багатьох різних програм і не є спеціалізованою для будь-яких конкретних завдань. Мова Python стала основною у науці про дані, дозволяючи аналітикам даних та іншим фахівцям використовувати цю мову для проведення складних статистичних розрахунків, створення візуалізації даних, побудови алгоритмів машинного навчання, аналізу даних та виконання інших завдань, пов'язаних з даними.

Python дозволяє створювати широкий спектр різних візуалізацій даних, таких як лінійні та стовпчасті діаграми, кругові діаграми, гістограми та тривимірні графіки. У Python також є ряд бібліотек, які дозволяють програмістам швидше та ефективніше писати програми для аналізу даних та машинного навчання, наприклад, TensorFlow та Keras [18].

Keras — це бібліотека API для нейронних мереж, написана на мові Python. Це бібліотека з відкритим кодом, яка працює поверх фреймворків, таких як Theano і TensorFlow. Створена для модульного, швидкого і зручного використання, Keras була створена інженером Google Франсуа Шолле. Він пропонує простий і інтуїтивний спосіб створення моделей глибинного навчання. На сьогодні, Keras є однією з найбільш широко використовуваних бібліотек API для розробки і тестування нейронних мереж, з якою дуже легко створювати шари для нейронних мереж або конфігурувати складні архітектури.

Модель Keras складається з послідовності або незалежного графа. Існує кілька повністю налаштованих модулів, які можуть бути об'єднані для створення нових моделей. Однією з переваг цієї модульності є те, що дуже легко додавати

нові функціональні можливості у вигляді окремих модулів. Можливе об'єднання модулів нейронних шарів, оптимізаторів, схем ініціалізації, функцій активації або схем регуляризації для створення нових модулів. Нові модулі дуже легко додавати, як і нові класи і функції. Шаблони визначаються в коді Python, а не в окремих файлах налаштування шаблонів [20].

TensorFlow – це бібліотека Python для швидких числових обчислень, створена і опублікована компанією Google. Вона також підтримує традиційне машинне навчання. Спочатку TensorFlow була розроблена для великих чисельних обчислень без урахування глибокого навчання. Однак вона виявилася дуже корисною для розробки глибокого навчання, тому Google виклав її у відкритий доступ. TensorFlow приймає дані як багатомірних масивів вищої розмірності, званих тензорами. Багатовимірні масиви дуже зручні під час роботи з великими обсягами даних. TensorFlow працює на основі графів потоків даних, які мають вузли та ребра. Оскільки механізм виконання має форму графів, набагато простіше виконувати код TensorFlow розподіленим чином кластері комп'ютерів під час використання графічних процесорів.

NumPy (Numerical Python) – це бібліотека Python з відкритим кодом, яка використовується практично у всіх галузях науки і техніки. Вона є універсальним стандартом для числових даних мовою Python, і є ядром наукових екосистем Python і PyData. Бібліотека NumPy містить структури даних багатовимірних масивів і матриць. Вона може бути використана для виконання широкого спектру математичних операцій над масивами. Додає структури даних в Python, які гарантують ефективне обчислення з масивами і матрицями, і забезпечує величезну бібліотеку високорівневих математичних функцій, які працюють з цими масивами і матрицями [21].

3.2. Бінарна класифікація

У цьому розділі описано послідовність розробки нейронної мережі та її навчання для вирішення задачі бінарної класифікації текстових даних.

Бінарна класифікація – це один із типів завдань класифікації у машинному навчанні, коли ми повинні класифікувати два взаємовиключні класи.

Будемо використовувати набір даних IMDB Dataset of 50K Movie Reviews [22], який містить 50 тисяч відгуків на фільми. Вони поділяються на 25000 відгуків для навчання та тестування. Кожен набір містить рівну кількість (50%) позитивних і негативних відгуків. Необхідно за текстом відгуку визначити, є відгук позитивним чи негативним. Для розроблення та тестування нейронної мережі був обрана саме цей набір даних, тому що він є досить популярним серед дослідників та розробників, що дає можливість порівняти отримані нами результати з вже проведеними дослідженнями та експериментами показниками точності розпізнавання тональності текстових даних.

3.2.1. Опис розробки програми

В якості мови програмування був обраний Python, тому що він найкраще підходить для машинного навчання (ML) та проектів на базі ШІ. Простота, послідовність, доступ до бібліотек і фреймворків для ШІ та машинного навчання, гнучкість, незалежність від платформи та широка спільнота додає мові загальної популярності.

Під час розробки та експериментів використовувався Google Colaboratory, який дозволяє будь-кому писати та виконувати довільний код Python через браузер, і особливо добре підходить для машинного навчання, аналізу даних та освіти.

Підключимо Google диск для того, щоб можна було працювати з файлами, які на ньому зберігаються (рис. 3.1).

```
[2] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Рисунок 3.1 – Підключення файлів

Завантажимо набір даних IMDB Dataset та надрукуємо декілька перших записів досліджуваного набору [22] (рис. 3.2).

```
[ ]  import pandas as pd
     import numpy as np

     df = pd.read_csv('drive/MyDrive/Collab/movie_data.csv', engine='python', quoting = 1, sep=',')
     df.head()
```

	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0
3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0

Рисунок 3.2 – Перші п'ять відгуків набору даних

Імпортуємо модулі для роботи із нейронними мережами (рис. 3.3):

– модуль array з NumPy буде використовуватися для конвертації датасету в масиви типу NumPy;
– one_hot з Keras – для кодування слів масивами цілих чисел;
– pad_sequences будуть використані для вирівнювання векторів речень до однакової довжини;
– pandas – для завантаження файлу із .csv;
– Sequential – для побудови послідовної моделі нейронної мережі;
– Dense – для додавання повнозв'язних шарів;
– Flatten – для зміни розмірностей масиву;
– Embedding – для реалізації шару word embedding.

```
from numpy import array
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers.embeddings import Embedding
```

Рисунок 3.3 – Імпорт необхідних модулів

Створюємо змінні для зберігання оглядів фільмів та міток (рис. 3.4).

```
[5]  docs=df['Review']
     labels=array(df['Status'])
```

Рисунок 3.4 – Змінні для зберігання оглядів фільмів та міток

Зафіксуємо для numpy та tensorflow випадкову ініціалізацію та переконаймося у стабільності отриманого результату (рис. 3.5.).

```
from numpy.random import seed
seed(42)
import tensorflow as tf
tf.compat.v1.set_random_seed(42)
```

Рисунок 3.5 – Фіксація випадкової ініціалізації

Використаємо вбудовану функцію зі sklearn для розбиття датасету на навчальний (80%) та тестовий набори (20%) (рис. 3.6). Надрукуємо для прикладу перший відгук навчального набору (рис. 3.7).

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(docs,labels,test_size=0.20)
```

Рисунок 3.6 – Вбудована функція зі sklearn для розбиття дата сету

```
[ ] print(X_train[1])

OK... so... I really like Kris Kristofferson and his usual easy going delivery of lines in his movies. Age has helped him with his soft spoken lo
```

Рисунок 3.7 – Перше речення із навчального набору

Більшість існуючих алгоритмів машинного навчання не можуть бути застосовані до категорійних даних. Категорійні дані необхідно спочатку перетворити в числові. На наступному кроці ми конвертуємо кожне речення в набір чисел (рис. 3.8) із використанням функції one_hot, яка зазвичай використовується в задачах послідовної класифікації. Вона має наступні аргументи: текст для конвертації; розмір словника; фільтр, який відкидає з результатів знаки пунктуації; булеве значення, яке вказує, чи потрібно трансформувати всі великі літери в малі; split вказує розділовий знак між словами.

```
[94] vocab_size= 15000
     X_train=[one_hot(d,vocab_size,filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~',lower=True,split=' ')for d in X_train]
     X_test=[one_hot(d,vocab_size,filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~',lower=True,split=' ')for d in X_test]
```

Рисунок 3.8 – Конвертація речень в набори чисел

Подивимось на результат (рис. 3.9).

```
[95] print(X_train[1])

     [8000, 8150, 1928, 1928, 2150, 767, 6119, 9033, 9402, 8836, 9033, 6663, 12922, 9432, 1078, 4430, 2666, 14660, 14935, 4048, 2666, 14163, 7574,
```

Рисунок 3.9 – Результат конвертації першого речення із навчального набору

Якщо подивитись на інше речення в числовому вигляді, то воно буде мати іншу довжину. Традиційно нейронні мережі будуються таким чином, що всі вхідні вектори (так само, як і вектори, які потім будуть використовуватися для класифікації) мають однакову довжину. Функція pad_sequences дозволяє вирівняти всі чисельні вектори речень в тренувальному наборі до однакової довжини (рис. 3.10). Ця функція в якості параметрів отримує масив значень (в нашому випадку – масив векторів чисел, кожне з яких відповідає слову), максимально дозволену його довжину та спосіб додавання нулів, в нашому випадку – на початку (рис.3.11).

```
[96] max_length=2200
     X_train=pad_sequences(X_train,maxlen=max_length,padding='pre')
     X_test=pad_sequences(X_test,maxlen=max_length,padding='pre')
```

Рисунок 3.10 – Вирівнювання всіх чисельних векторів речень

```
[97] print(X_train[1])

     [   0    0    0 ... 5350 4430 6399]
```

Рисунок 3.11 – Додавання необхідної кількості нулів у початок векторів

Тепер можна будувати модель нейронної мережі. Це буде мережа прямого розповсюдження, тож використаємо модель Sequential. Наступний шар зветься Embedding. Перший параметр у шарі вбудовування – це розмір словника або загальна кількість унікальних слів у корпусі. Другий параметр – це кількість вимірів кожного вектора слів. Наприклад, якщо ви хочете, щоб кожен вектор слів мав 32 виміри, ви вкажете 32 як другий параметр. Третій параметр – це довжина вхідного речення. Результатом вбудовування слів є двовимірний вектор, у якому слова представлені рядками, а відповідні їм розміри стовпцями .Цей шар будує

вектор embedding, який визначає унікальність кожного текстового речення та їх близькість. Ці вектори будуть змінюватися під час навчання.

Щоб з'єднати шар вбудовування слів із щільно пов'язаним шаром, потрібно двовимірний вихід з шару Embeddings "сплющити" в одновимірний масив за допомогою Flatten та передається на два послідовні повнозв'язні шари Dense із відповідними функціями активації relu та sigmoid. Між ними розташовано шар відкидання певної частини нейронів (Dropout) для запобігання перенавчанню. Після створення моделі вона компілюється (compile) із відповідним методом оптимізації (було обрано Adam), функцією втрат (binary_crossentropy) та метрикою оцінки якості (metrics). В наступному рядку вказано метод summary, який друкує структуру всієї моделі. Тепер можна запускати навчання (fit). При навчанні вказується навчальна вибірка з двох частин X_train, y_train, кількість ітерацій, коефіцієнт розбиття між навчальною та валідаційною вибіркою. Додатково можна параметром verbose вказати рівень деталізації друку результатів (рис. 3.12, 3.13).

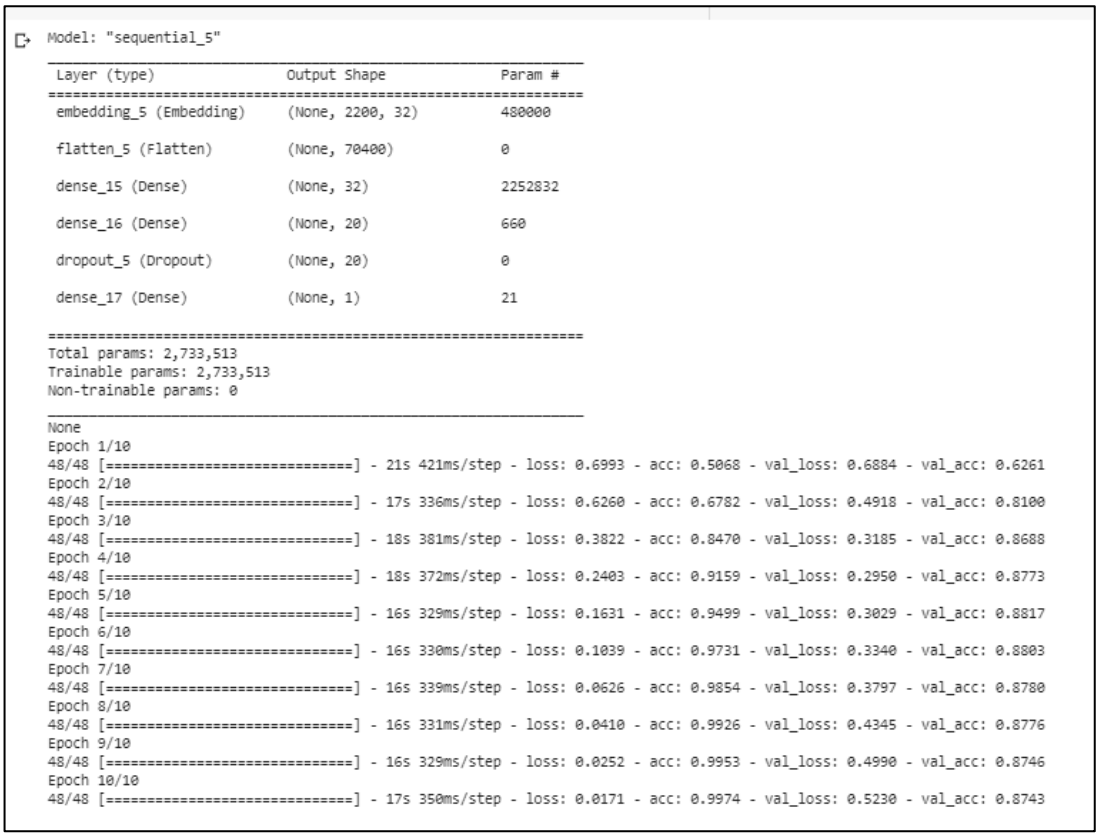
```
model=Sequential()
model.add(Embedding(vocab_size,32,input_length=max_length))
model.add(Flatten())
model.add(Dense(32,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

print(model.summary())

history=model.fit(X_train,y_train,epochs=10,verbose=1,validation_split=0.4, batch_size=500)
```

Рисунок 3.12 – Побудова, компіляція, виведення та навчання моделі



```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 2200, 32)          480000

 flatten_5 (Flatten)         (None, 70400)             0

 dense_15 (Dense)            (None, 32)                2252832

 dense_16 (Dense)            (None, 20)                660

 dropout_5 (Dropout)         (None, 20)                0

 dense_17 (Dense)            (None, 1)                 21

=================================================================
Total params: 2,733,513
Trainable params: 2,733,513
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/10
48/48 [==============================] - 21s 421ms/step - loss: 0.6993 - acc: 0.5068 - val_loss: 0.6884 - val_acc: 0.6261
Epoch 2/10
48/48 [==============================] - 17s 336ms/step - loss: 0.6260 - acc: 0.6782 - val_loss: 0.4918 - val_acc: 0.8100
Epoch 3/10
48/48 [==============================] - 18s 381ms/step - loss: 0.3822 - acc: 0.8470 - val_loss: 0.3185 - val_acc: 0.8688
Epoch 4/10
48/48 [==============================] - 18s 372ms/step - loss: 0.2403 - acc: 0.9159 - val_loss: 0.2950 - val_acc: 0.8773
Epoch 5/10
48/48 [==============================] - 16s 329ms/step - loss: 0.1631 - acc: 0.9499 - val_loss: 0.3029 - val_acc: 0.8817
Epoch 6/10
48/48 [==============================] - 16s 330ms/step - loss: 0.1039 - acc: 0.9731 - val_loss: 0.3340 - val_acc: 0.8803
Epoch 7/10
48/48 [==============================] - 16s 339ms/step - loss: 0.0626 - acc: 0.9854 - val_loss: 0.3797 - val_acc: 0.8780
Epoch 8/10
48/48 [==============================] - 16s 331ms/step - loss: 0.0410 - acc: 0.9926 - val_loss: 0.4345 - val_acc: 0.8776
Epoch 9/10
48/48 [==============================] - 16s 329ms/step - loss: 0.0252 - acc: 0.9953 - val_loss: 0.4990 - val_acc: 0.8746
Epoch 10/10
48/48 [==============================] - 17s 350ms/step - loss: 0.0171 - acc: 0.9974 - val_loss: 0.5230 - val_acc: 0.8743
```

Рисунок 3.13 – Процес побудови та навчання моделі

Після завершення процедури навчання за допомогою evaluate можна оцінити якість моделі на навчальній та тестовій вибірці (рис. 3.14, 3.15).

```
[99] loss,accuracy=model.evaluate(X_train,y_train,verbose=1)
     print('Training accuracy is {}'.format(accuracy*100))

     1250/1250 [==============================] - 12s 9ms/step - loss: 0.2113 - acc: 0.9495
     Training accuracy is 94.94749903678894
```

Рисунок 3.14 – Якість моделі на навчальній вибірці

```
[100] loss,accuracy=model.evaluate(X_test,y_test)
      print('Testing accuracy is {}'.format(accuracy*100))

      313/313 [==============================] - 3s 9ms/step - loss: 0.5075 - acc: 0.8778
      Testing accuracy is 87.77999877929688
```

Рисунок 3.15 – Якість моделі на тестовій вибірці

Точність (accuracy) навчальної вибірки склала майже 95%, а як ми бачимо точність на тестовій вибірці є дещо гіршою, ніж на навчальній, та складає 87,78%.

Порівняємо результати класифікації розробленої нами нейронної мережі з дослідженнями, які проводилися науковцями в статтях [23, 24] для того ж самого набору даних IMDB Dataset. В статті [24] точність класифікації наївним байєсовим класифікатором склала 81%, класифікатором SVM 82,9%. В статті [23] CNN дав точність 87,7%, у той час як MLP та LSTM дали точність 86,74% та 86,64 відповідно. Тобто можна сказати, що результат точності, який ми отримали при проектуванні власної нейронної мережі є трохи вищим за класифікацію текстових даних за настроєм, які були опубліковані в інших роботах із використанням наборів даних IMDB Dataset англійською мовою.

На наступному кроці побудуємо графіки навчання для аналізу та візуалізації процесу навчання (рис. 3.16, 3.17).

```
[17] import matplotlib.pyplot as plt

     def plot_graphs (history,metric):
       plt.plot (history.history[metric])
       plt.plot (history.history['val_'+metric], '')
       plt.xlabel("Epochs")
       plt.ylabel(metric)
       plt.legend([metric, 'val_'+ metric])

     plt.figure(figsize=(16,8))
     plt.subplot(1,2,1)
     plot_graphs (history,'acc')
     plt.ylim(None,1)
     plt.subplot (1,2,2)
     plot_graphs (history,'loss')
     plt.ylim(0,None)
```

Рисунок 3.16 – Побудова графіків навчання

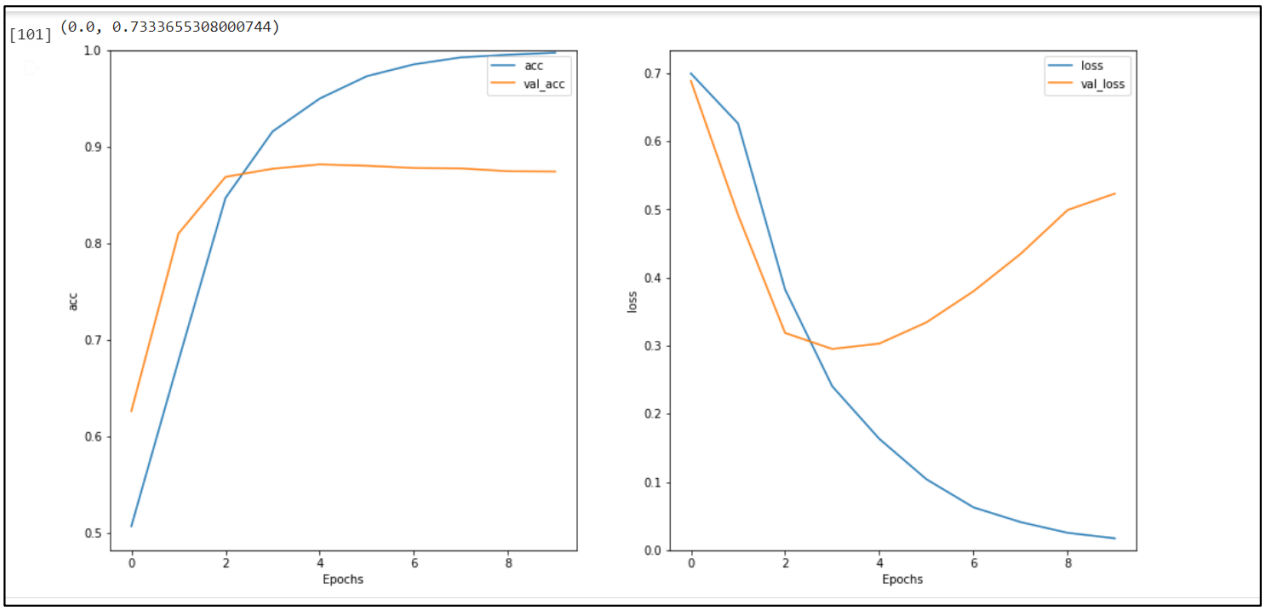


Рисунок 3.17 – Побудовані графіки навчання

На графіку праворуч відображаються значення помилки навчання та ліворуч точність навчання для окремих ітерацій навчальної та валідаційної вибірки. Відхилення тренувального набору зменшується з кожною ітерацією (епохою), а точність навчання збільшується з кожною епохою. Це очікувано при використанні оптимізації градієнтного спуску — вона повинна мінімізувати значення помилки на кожній ітерації. Але це не стосується похибки і точності для валідаційної. Це є прикладом перенавчання (overfitting): модель працює набагато краще з навчальними даними, ніж з даними, яких вона ніколи не бачила. В процесі такого навчання з певного моменту (зростання сумарної помилки val_loss) модель починає надмірно оптимізувати навчальний набір, але не узагальнюється для довільних векторів. У нашому випадку щоб запобігти такому перенавчанню, ми зупинимо навчання вчасно, а також спробуємо додати ще один шар відкидання певної частини нейронів (Dropout) після першого повнозв'язного Dense шару.

Оцінимо класифікацію довільних речень. Перший відгук є позитивним, результат його класифікації дорівнює 0,554, що ближче до 1 – класу позитивних відгуків, ніж до 0 – класу негативних (рис. 3.18).

```
[108] sample_text=('Saw the movie today and thought it was a good effort, good messages for kids.')
      oh=one_hot(sample_text,vocab_size,filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~',lower=True,split=' ')
      print(oh)
      pad=pad_sequences([oh],maxlen=max_length,padding='pre')
      predictions=model.predict(pad)
      print(predictions)

      [10212, 2666, 5192, 2336, 1853, 8909, 9409, 6642, 9533, 6944, 6965, 6944, 12349, 9324, 4814]
      [[0.55447656]]
```

Рисунок 3.18 – Результат класифікації позитивного відгуку

Другий відгук є негативним, відповідно, результат його класифікації дорівнює 0,001 (рис. 3.19).

```
[109] sample_text=('The acting was bad, the dialogs were extremely shallow and insincere.')
      oh=one_hot(sample_text,vocab_size,filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~',lower=True,split=' ')
      print(oh)
      pad=pad_sequences([oh],maxlen=max_length,padding='pre')
      predictions=model.predict(pad)
      print(predictions)

      [2666, 12255, 6642, 10707, 2666, 5818, 5128, 3013, 9464, 1853, 7273]
      [[0.00150684]]
```

Рисунок 3.19 – Результат класифікації негативного відгуку

3.2.2 Вихідні дані для навчання нейронної мережі з класифікації україномовних новин

Для успішного навчання нейронної мережі потрібно дуже великий обсяг даних. Ми будемо використовувати статті з сайту tsn.ua – це лідер новин в українському Інтернеті. ТСН – це найсвіжіші та найоперативніші новини України та світу 24 години на добу та 7 днів на тиждень.

Для розробки алгоритму класифікації статей по категоріям (політика, спорт, економіка тощо) нам потрібно провести парсінг сайту за допомогою Python з використанням бібліотек beautifulsoup та requests.

Відкриваємо Google Colab, імпортуємо бібліотеки з допомогою import: requests, BeautifulSoup, pandas. (рис. 3.20.).

```
[ ]  import requests
     from bs4 import BeautifulSoup
     import pandas as pd
```

Рисунок 3.20 – Імпорт бібліотек

Щоб звернутися до сайту, створюємо змінну url та привласнюємо їй значення сайту ТСН. Відправимо на сервер запит за допомогою команди get, на вхід якої передаємо змінну url. За допомогою команди text подивимося, що вийшло (рис. 3.21).

```
url = "https://tsn.ua/tags/харків"
r = requests.get(url)
r.text
```

```
'<!DOCTYPE html><html lang="uk" class="js-focus-visible"><head><meta charset="utf-8"><meta http-equiv="x-ua-compatible" content="ie=edge"><meta n
ame="viewport" content="viewport-fit=cover, initial-scale=1, shrink-to-fit=no, width=device-width"><meta property="fb:app_id" content="1042761182
504521" /><link rel="preconnect" href="https://ls.hit.gemius.pl" crossorigin><link rel="dns-prefetch" href="https://ls.hit.gemius.pl" crossorigin
><link rel="preconnect" href="https://www.gstatic.com" crossorigin><link rel="dns-prefetch" href="https://www.gstatic.com" crossorigin><link rel
="preconnect" href="https://graph.facebook.com" crossorigin><link rel="dns-prefetch" href="https://graph.facebook.com" crossorigin><link rel="pre
connect" href="https://securepubads.g.doubleclick.net" crossorigin><link rel="dns-prefetch" href="https://securepubads.g.doubleclick.net" crossor
igin><link rel="preconnect" href="https://membrana-cdn.media/" crossorigin><link rel="dns-prefetch" href="https://membrana-cdn.media…'
```

Рисунок 3.21 – Код сайту

Створимо змінну soup, передамо r.text і вкажемо необхідний формат (рис. 3.22).

```
soup = BeautifulSoup (r.text, 'lxml')
```

```
soup
```

```
<!DOCTYPE html>
<html class="js-focus-visible" lang="uk"><head><meta charset="utf-8"/><meta content="ie=edge" http-equiv="x-ua-compatible"/><meta content="viewp
window.tsn = {"root":"https:\/\/tsn.ua","url":"https:\/\/tsn.ua\/tags\/%D1%85%D0%B0%D1%80%D0%BA%D1%96%D0%B2","environment":"production","videoR
window.tsn.staticAsset = function (path) {return "https://tsn.ua/{path}?v=632".replace('{path}', path)}
window.tsn.addClosure = closure => {return window.tsn.closures.push(closure)}
window.tsn.adTargeting = {"PageType_Tsn":["Tag"],"PageType_TagTsn":["xarkiv"]}
</script><script type="text/javascript">
    var _paq = _paq || [];
    _paq.push(["setDomains", ["*.tsn.ua", "tsn.ua", "*.dev", "*.test"]]);
    _paq.push(['trackPageView']);
    _paq.push(['enableLinkTracking']);
    (function() {
        var u="//assay.tsn.ua/";
        _paq.push(['setTrackerUrl', u+'piwik.php']);
         _paq.push(['setSiteId', '1']);
        var d=document, g=d.createElement('script'), s=d.getElementsByTagName('script')[0];
        g.type='text/javascript'; g.async=true; g.defer=true; g.src=u+'piwik.js'; s.parentNode.insertBefore(g,s);
    })();
</script><script>
```

Рисунок 3.22 – Упорядкований код сайту

Створимо змінну data, в яку додаватимемо всі змінні, які запитуємо.

Зберемо дані зі сторінок. Створимо перед циклом змінну url і скористаємося методом f-рядків, щоб помістити всередину фігурних дужок той параметр, який нам потрібен – номер сторінки. Назвемо параметр p та помістимо його в цикл у проміжку від 1 до 6 для того, щоб зібрати інформацію з перших 6 сторінок.

Створимо змінну news, задамо пошук по першому тегу div і класу c-card__body, з командою find_all.

Напишемо цикл для кожного new у списку news (рис. 3.23).

```
data = []

for p in range(1, 6):

  url = f"https://tsn.ua/tags/харків/page-{p}"
  r = requests.get(url)
  soup = BeautifulSoup (r.text, 'lxml')

  news = soup.find_all('div',class_='c-card__body')

  for new in news:
    link = new.find('a',class_='c-card__link').get('href')
    name = new.find('h3',class_="c-card__title").text.split('\n')
    data.append([name,link])
```

Рисунок 3.23 – Збір даних

Запустимо код і подивимося, що відображається в data (рис. 3.24).

```
data

[[['Харківщина вкотре під вогнем: окупанти обстріляли житлові будинки та поранили літніх людей'],
  'https://tsn.ua/ato/harkivschina-vkotre-pid-vognem-okupanti-obstrilyali-zhitlovi-budinki-ta-poranili-litnih-lyudey-2086642.html'],
 [['Харків знову опинився в зоні досяжності ствольної артилерії супротивника - Арестович'],
  'https://tsn.ua/ato/harkiv-znovu-opinivsya-v-zoni-dosyazhnosti-stvolnoyi-artileriyi-suprotivnika-arestovich-2085913.html'],
 [['На Харківщині під час обстрілів загинула мати, її 3-місячну дитину - поранено'],
  'https://tsn.ua/ukrayina/na-harkivschini-pid-chas-obstriliv-zaginula-mati-yiyi-3-misyachnu-ditinu-poraneno-2086114.html'],
 [['Російські окупанти гатили по Харкову касетними бомбами - Amnesty International\xa0'],
  'https://tsn.ua/ato/rosiyski-okupanti-gatili-po-harkovu-kasetnimi-bombami-amnesty-international-2085040.html'],
 [['Російські окупанти завдяки штурму закріпилися на околиці Ізбицького на Харківщині - Генштаб'],
  'https://tsn.ua/ukrayina/rosiyski-okupanti-zavdyaki-shturmu-zakripilisya-na-okolici-izbickogo-na-harkivschini-genshtab-2085016.html'],
 [['Окупанти обстріляли населенні пункти Харківщини: є загиблі серед цивільних'],
  'https://tsn.ua/ukrayina/okupanti-obstrilyali-naselenni-punkti-harkivschini-ye-zagibli-sered-civilnih-2084587.html'],
 [['Окупанти б’ють по Харкову бомбами та ракетами більшої потужності: Терехов описав ситуацію в місті'],
  'https://tsn.ua/ato/okupanti-b-yut-po-harkovu-bombami-ta-raketami-bilshoyi-potuzhnosti-terehov-opisav-situaciyu-v-misti-2083501.html'],
 [['"Чесність є зброєю проти усього, що несе Росія": Зеленський – про відновлення українського телевізійного мовлення у Харкові'],
  'https://tsn.ua/ato/chesnist-ye-zbroyeyu-proti-usogo-scho-nese-rosiya-zelenskiy-pro-vidnovlennya-ukrayinskogo-televiziynogo-movlennya-u-harko
 [['"Своєрідна помста з боку Росії: військовий експерт - про збільшення обстрілів Харкова'],
  'https://tsn.ua/ato/svoyeridna-pomsta-z-boku-rosiyi-viyskoviy-ekspert-pro-zbilshennya-obstriliv-harkova-2082772.html'],
```

Рисунок 3.24 – Відображення даних в data

Введемо назви для колонок – нехай вони дублюють назви змінних, які ми привласнили раніше. Після цього імпортуємо дані в csv файл (рис. 3.25).

```
[ ]  header = ['name','link']

[ ]  df = pd.DataFrame(data, columns = header)
     df.to_csv('news_data.txt',sep=';', encoding = 'utf8')
```

Рисунок 3.25 – Імпорт даних в csv таблицю

3.2.3 Бінарна класифікація україномовних новин

Дослідимо можливість використання створеної нами нейронної мережі для бінарної класифікації україномовних заголовків статей. Допустімо, ми ведемо розділ політичних новин та хочемо оперативно визначати за аналізом заголовку, чи належить нова стаття нашій рубриці чи ні. Натренуймо нашу мережу на вже існуючих новинах. Нас цікавить категорія номер 3, тож привласнимо 1 тим новинам, які відповідають 3 категорії, а іншим дамо мітку 0.

Загальна кількість новин 3 категорії складає 4136, а загальна кількість початкового набору даних складає 24765 новин.

Завантажуємо необхідні бібліотеки, зчитуємо з файлів вихідну інформацію, розбиваємо датасет на навчальний (80%) та тестовий набори (20%). На наступному кроці ми конвертуємо кожне речення в набір чисел із використанням функції one_hot. Вирівнюємо довжину речень. Будуємо та навчаємо нейронну мережу (рис. 3.26). Після завершення процедури навчання можна оцінити якість моделі на навчальній та тестовій вибірці (рис. 3.27).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 33, 32)            1436160

 flatten (Flatten)           (None, 1056)              0

 dense (Dense)               (None, 32)                33824

 dense_1 (Dense)             (None, 20)                660

 dropout (Dropout)           (None, 20)                0

 dense_2 (Dense)             (None, 1)                 21

=================================================================
Total params: 1,470,665
Trainable params: 1,470,665
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/50
372/372 [==============================] - 7s 17ms/step - loss: 0.4826 - acc: 0.8298 - val_loss: 0.4444 - val_acc: 0.8372
Epoch 2/50
372/372 [==============================] - 6s 16ms/step - loss: 0.3626 - acc: 0.8486 - val_loss: 0.5373 - val_acc: 0.7787
Epoch 3/50
372/372 [==============================] - 6s 17ms/step - loss: 0.0845 - acc: 0.9695 - val_loss: 0.8524 - val_acc: 0.7191
Epoch 4/50
372/372 [==============================] - 6s 16ms/step - loss: 0.0264 - acc: 0.9865 - val_loss: 1.1161 - val_acc: 0.7685
Epoch 5/50
```

Рисунок 3.26 – Процес побудови та навчання моделі

```
620/620 [==============================] - 1s 2ms/step - loss: 0.4384 - acc: 0.8339
Training accuracy is 83.38885307312012
155/155 [==============================] - 0s 2ms/step - loss: 0.4571 - acc: 0.8294
Testing accuracy is 82.93963074684143
```

Рисунок 3.27 – Якість моделі на навчальній та тестовій вибірці.

Як бачимо, показник точності моделі набагато нижчий, ніж при розпізнаванні тональності тексту, але прийнятний як для класу подібних задач. Проаналізувавши заголовки новин, ми дійшли висновків, що такий результат, скоріш за все, обумовлений схожістю заголовків новин із різних категорій, що відображено у хмарі слів (рис. 3.28).

Рисунок 3.28 – Хмари слів новин 3 та 5 категорій

На рисунку ліворуч розташована хмара слів з 3 категорії, а праворуч з 5. Як бачимо різниця між ними для людини-оператора зовсім не очевидна.

3.3 Оптимізація мережі

Нейронні мережі забезпечують досить високу точність, тому широко поширені для вирішення різних класів задачах. Але часто точність мережі, яку ми будуємо, може бути не задовільною або не давати бажані результати. Тому ми завжди шукаємо способи покращення продуктивності моделі. Існують різні методи для досягнення кращих результатів [25].

Перший крок у тому, щоб переконатися, що нейронна мережа добре працює з тестовими даними – це переконатися, що нейронна мережа не перенавчається. Перенавчання відбувається, коли модель починає запам’ятовувати значення з навчальних даних замість того, щоб вчитися на них. Тому, коли модель стикається з даними, яких вона раніше не бачила, вона не зможе добре працювати з ними.

Якщо точність навчання набагато вища, ніж точність тестування, можна стверджувати, що модель перенавчається. Також можна побудувати графік процесу навчання, щоб перевірити. Є кілька прийомів, щоб уникнути перенавчання. Перше це регуляризація даних. Другий метод – викидання – випадкове відключення зв’язків між нейронами, що змушує мережу шукати нові шляхи та узагальнюватися. Рання зупинка – прискорює навчання нейронної мережі, що призводить до зменшення помилок у тестовому наборі.

Другий крок в оптимізації мережі – гіперпараметри. Це значення, які ви повинні ініціалізувати в мережі, наприклад, у згортковій нейронній мережі деякі гіперпараметри – це розмір ядра, кількість шарів у нейронній мережі, функція активації, функція втрат, оптимізатор, batch size, кількість епох для навчання тощо. На жаль, немає прямого методу визначення найкращого набору гіперпараметрів для кожної нейронної мережі, тому його в основному отримують

шляхом проб і помилок. Але є декілька основних параметрів, на які потрібно звернути особливу увагу.

Швидкість навчання – вибір оптимальної швидкості навчання важливий, оскільки він вирішує, чи зближається ваша мережа до глобальних мінімумів чи ні.

Архітектура мережі – не існує стандартної архітектури, яка б забезпечувала високу точність у всіх тестових випадках. Потрібно експериментувати, випробовувати різні архітектури, отримувати висновки з результату і повторювати спроби. Також краще використовувати перевірені архітектури замість того, щоб будувати свою власну мережу.

Функція оптимізаторів (optimizer) і втрат (loss) також може бути обрана з безлічі варіантів. Найчастіше використовуються оптимізатори RMSprop, Stochastic Gradient Descent і Adam.

Розмір пакету (batch size) та кількість епох (epochs). Batch size. Пакет або міні-пакет – невеликий набір зразків, оброблюваних моделлю одночасно. У процесі навчання один міні-пакет використовується в градієнтному спуску для обчислення однієї зміни ваги моделі. Не існує стандартного значення для розміру пакету та епох, яке працює для всіх випадків використання. Треба експериментувати, але у загальній практиці значення розміру пакету встановлюються як 8, 16, 32 тощо.

Варіювання параметра Validation split є одним з варіантів боротьби з перенавчанням мережі.

Функція активації – функції активації відображають нелінійні функціональні входи і виходи. Функції активації дуже важливі, і вибір правильної функції активації допомагає моделі краще навчатися. На сьогоднішній день Rectified Linear Unit (ReLU) є найбільш широко використовуваною функцією активації.

Зафіксуємо такі початкові параметри: максимальну довжину, як найбільшу довжину речень в базі текстів (2332) та знайдемо обсяг словника, що складає 340252 слів. Будемо змінювати розмір вектора в шарі Embedding, кількість шарів

Dense та кількість нейронів в шарі, а також максимальну кількість слів в реченні та обсяг словника. Результати експериментів відображено в табл. 3.1.

Таблиця 3.1 – Порівняння архітектур нейронних мереж

Схема мережі	**Інші параметри**	**Точність (accuracy), %**
Embedding (32)+Dense(20)+Dense(20)		89,3
Embedding (32)+Dense(15)+Dense(15)		89,3
Embedding (64)+Dense(64)		89,54
Embedding (32)+Dense(20)+Dense(20)	max_len =1200	88,95
Embedding (32)+Dense(20)+Dense(20)	max_len =2332	89,25
Embedding (32)+Dense(20)+Dense(20)	max_len =2332, vocab_size= 15000	89,25
Embedding (32)+Dense(32)+Dense(20)	max_len =2200 vocab_size= 15000	87,97
Embedding (32)+Dense(32)+Dense(20)	max_len =2332 vocab_size= 340252	89,49
Embedding (168)+Dense(32)+Dense(20)	batch_size=254	89,34
Embedding (168)+Dense(32)+Dense(20)	batch_size=128	89,17
Embedding (168)+Dense(32)+Dense(20)	batch_size=64	88,94
Embedding (32)+Dense(32)+ Dropout(0.3)+Dense(20)	batch_size=256	88,98
Embedding (32)+Dense(32)+Dense(20)	max_len =2332 vocab_size= 340252 batch_size=256	89,62

Найкращі показники ми отримали в останній моделі, структуру якої пропонується залишити для подальших експериментів.

Також для обраної моделі мережі проведені дослідження параметра Validation split для двох розмірів пакетів (batch size), які дорівнювали 256 та 512. Результати експериментів відображено в табл. 3.3.

Таблиця 3.2 – Дослідження параметра Validation split

Validation split =	**Точність (accuracy), %**	
	batch size = 256	**batch size = 512**
0,1	89,46	89,23
0,2	89,31	89,20
0,3	89,09	89,14
0,4	89,25	89,19
0,5	88,74	89,06

З табл. 3.2 можна побачити, що найкращі результати отримані для batch size = 256 та Validation split = 0,1. Ці гіперпараметри вирішено залишити для розробленої мережі.

Також для збільшення точності навчання можна спробувати покращити попередню обробку текстових даних. Одним із методів є видалення пунктуації, приведення тексту до нижнього регістру (рис. 3.29), аналіз хмари слів вихідних текстів та видалення стоп-слів, які не несуть цінного змісту та не спливають на результат класифікації, застосування стеммінгу.

```
[31] import re

     df['text_processed'] = df['review'].map(lambda x:re.sub('[,\.!?]','',x))

     df['text_processed'] = df['text_processed'].map(lambda x:x.lower())

     df['text_processed'].head()

     0    in 1974 the teenager martha moxley (maggie gra...
     1    ok so i really like kris kristofferson and his...
     2    ***spoiler*** do not read this if you think ab...
     3    hi for all the people who have seen this wonde...
     4    i recently bought the dvd forgetting just how ...
     Name: text_processed, dtype: object
```

Рисунок 3.29 – Видалення пунктуації та приведення тексту до нижнього регістру

Стоп-слова – це тривіальні слова, такі як "I", "the", "you" и т.д., тощо, які з’являються в тексті настільки часто, що можуть викривити результати NLP. Тому майже завжди треба видаляти стоп-слова з корпусу в рамках попередньої обробки. Бібліотека NLTK містить список із приблизно 179 стоп-слів англійською мовою, який можна використовувати для фільтрації тексту (рис. 3.30).

```
import nltk
from nltk.stem.porter import PorterStemmer
import re, string
from tqdm import tqdm
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
nltk.download('stopwords')

def preprocessing(text):
  ps = PorterStemmer()
  text = [token for token in text.split() if len(token)>5 and (not token in set(stop_words))]
  text=[ps.stem(token) for token in text]
  text = [token for token in text if token]
  return ' '.join(text)
df['text_processed'] = df['text_processed'].apply(lambda x:preprocessing(x))
```

Рисунок 3.30 – Видалення стоп-слів та застосування стеммінгу

У NLTK є багато функцій стемінгу з різними алгоритмами, тут ми будемо використовувати алгоритм Портера (PorterStemmer) (рис. 3.30)

Після завершення процедури навчання за допомогою evaluate можна оцінити якість моделі на навчальній та тестовій вибірці. Як відмічалося на рис 3.14,3.15 початкова точність навчання є 94.94%, а точність тестування – 87.77%. Після додавання попередньої обробки та видалення стоп-слів точність навчання є 94.87%, а точність тестування – 87.84%. Як бачимо, суттєвих покращень точності навчання попередньою обробкою тексту нам досягти не вдалося.

3.4. Мультикласова класифікація новин.

У цьому розділі описано послідовність розробки нейронної мережі та її навчання для вирішення задачі мультикласової класифікації текстових даних.

Будемо використовувати набір даних статей BBC news [26], який містить 1 тисячу новин. Необхідно за текстом новин визначити, до якої з 6 категорій (розваги, бізнес, політика, спорт, техніка) відноситься новина.

Підключемо файли, які зберігаються на гул-диску. Завантажимо набір даних та надрукуємо декілька перших записів датасету (рис. 3.31).

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np

df = pd.read_csv('drive/MyDrive/Collab/BBC News Train - BBC News Train.csv', engine='python', quoting = 1, sep=',')
df.head()
```

	Category	Text
0	business	worldcom ex-boss launches defence lawyers defe...
1	business	german business confidence slides german busin...
2	business	bbc poll indicates economic gloom citizens in ...
3	tech	lifestyle governs mobile choice faster bett...
4	business	enron bosses in $168m payout eighteen former e...

Рисунок 3.31 – Перші п’ять записів датасету

Імпортуємо необхідні модулі для роботи із нейронними мережами (рис. 3.32).

```
import tensorflow as tf
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers import Activation
from keras.layers import Embedding
from keras.layers import Bidirectional
from keras.layers.embeddings import Embedding
```

Рисунок 3.32 – Модулі для роботи із нейронними мережами

Імпортуємо бібліотеку ntlk і функцію stopwords. Вкажемо стоп-слова для англійської мови (рис. 3.33).

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Рисунок 3.33 – Викидання стоп слів

Встановлюємо гіперпараметри. Ми задаємо гіперпараметри, необхідні для побудови і тренування моделі (рис. 3.34).

```
vocab_size = 5000
embedding_dim = 64
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = '<OOV>' # OOV = Out of Vocabulary
training_portion = .8
```

Рисунок 3.34 – Встановлення гіперпараметрів

Заповнюємо список статей і міток з даних, а також видаляємо стоп-слова. Отримаємо 1490 міток і 1490 – статей (рис. 3.35).

```
articles = []
labels = []

with open("drive/MyDrive/Collab/BBC News Train - BBC News Train.csv", 'r') as csvfile:
    df = csv.reader(csvfile, delimiter=',')
    next(df)
    for row in df:
        labels.append(row[0])
        article = row[1]
        for word in stop_words:
            token = ' ' + word + ' '
            article = article.replace(token, ' ')
            article = article.replace('  ', ' ')
        articles.append(article)
print(len(labels))
print(len(articles))
```

```
1490
1490
```

Рисунок 3.35 – Список статей та міток

Розбиття датасету на навчальну та валідаційну вибірку. Ми відводимо 80% текстових даних для навчальної вибірки і ще 20% для перевірки (рис. 3.36).

```
[ ] train_size = int(len(articles) * training_portion)

    train_articles = articles[0: train_size]
    train_labels = labels[0: train_size]

    validation_articles = articles[train_size:]
    validation_labels = labels[train_size:]

    print("train_size",  train_size)
    print(f"train_articles {len(train_articles)}")
    print("train_labels", len(train_labels))
    print("validation_articles", len(validation_articles))
    print("validation_labels", len(validation_labels))
```

```
train_size 1192
train_articles 1192
train_labels 1192
validation_articles 298
validation_labels 298
```

Рисунок 3.36 – Розбиття дата сету

Оскільки модель не розуміє слів, нам потрібно перетворити мітки на числа. Ми робимо токенизацію і перетворюємо речення на послідовність слів (рис. 3.37).

```
import numpy as np

label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(labels)

training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))
validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))
```

Рисунок 3.37 – Токенізація

Побудова, компіляція, виведення та навчання моделі нейронної мережі (рис. 3.38, 3.39).

```
model = Sequential()

model.add(Embedding(vocab_size, embedding_dim))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(embedding_dim)))
model.add(Dense(6, activation='softmax'))

model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, None, 64)          320000

 dropout_4 (Dropout)         (None, None, 64)          0

 bidirectional_4 (Bidirectio  (None, 128)              66048
 nal)

 dense_4 (Dense)             (None, 6)                 774

=================================================================
Total params: 386,822
Trainable params: 386,822
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.38 – Побудова моделі

```
num_epochs = 10
history = model.fit(train_padded, training_label_seq, epochs=num_epochs, validation_data=(validation_padded, validation_label_seq), verbose=2)
```

```
Epoch 1/10
38/38 - 14s - loss: 1.6632 - accuracy: 0.2559 - val_loss: 1.5515 - val_accuracy: 0.3054 - 14s/epoch - 373ms/step
Epoch 2/10
38/38 - 9s - loss: 1.3949 - accuracy: 0.4404 - val_loss: 1.5262 - val_accuracy: 0.4899 - 9s/epoch - 233ms/step
Epoch 3/10
38/38 - 9s - loss: 0.9753 - accuracy: 0.7114 - val_loss: 0.6367 - val_accuracy: 0.8188 - 9s/epoch - 248ms/step
Epoch 4/10
38/38 - 9s - loss: 0.5126 - accuracy: 0.8406 - val_loss: 0.5819 - val_accuracy: 0.8289 - 9s/epoch - 229ms/step
Epoch 5/10
38/38 - 9s - loss: 0.2860 - accuracy: 0.9161 - val_loss: 0.3767 - val_accuracy: 0.9060 - 9s/epoch - 233ms/step
Epoch 6/10
38/38 - 9s - loss: 0.1893 - accuracy: 0.9480 - val_loss: 0.3583 - val_accuracy: 0.8926 - 9s/epoch - 230ms/step
Epoch 7/10
38/38 - 9s - loss: 0.1551 - accuracy: 0.9673 - val_loss: 0.2598 - val_accuracy: 0.9161 - 9s/epoch - 233ms/step
Epoch 8/10
38/38 - 9s - loss: 0.0712 - accuracy: 0.9924 - val_loss: 0.2127 - val_accuracy: 0.9262 - 9s/epoch - 234ms/step
Epoch 9/10
38/38 - 9s - loss: 0.0520 - accuracy: 0.9908 - val_loss: 0.1901 - val_accuracy: 0.9463 - 9s/epoch - 233ms/step
Epoch 10/10
38/38 - 9s - loss: 0.0308 - accuracy: 0.9975 - val_loss: 0.1743 - val_accuracy: 0.9362 - 9s/epoch - 234ms/step
```

Рисунок 3.39 – Тренування моделі

Оцінимо класифікацію статей. Перша стаття відноситься до категорії бізнес. Друга стаття відноситься до 5 мітки – розваги (рис. 3.40, 3.41).

Рисунок 3.40 – Результат класифікації статті

Рисунок 3.41 – Результат класифікації статті

3.5 Практична цінність отриманих результатів

Текст може бути надзвичайно багатим джерелом інформації, але може бути складним і трудомістким витягнути корисну інформацію через її неструктуровану природу. Але завдяки досягненням у галузі обробки природної мови та машинного навчання, які відносяться до великого розділу штучного інтелекту, стає легше сортувати текстові дані. Вона працює шляхом автоматичного аналізу та структурування тексту, швидко та економічно, дозволяючи підприємствам автоматизувати процеси та отримувати інформацію, що призводить до більш ефективних рішень.

Класифікатори тексту можуть використовуватися для організації, структурування та категоризації практично будь-якого виду тексту – від

документів, медичних досліджень та файлів, а також у всьому Інтернеті. Наприклад, нові статті можуть бути організовані на теми; заявки до служби підтримки можуть бути організовані за терміновістю; розмови в чаті можуть бути організовані за мовою написання; згадки про бренд можуть бути організовані за настроєм і так далі.

Головну проблему, яку вирішує автоматизована класифікація даних – це людський фактор. По-перше ручний аналіз та класифікація – це повільно і менш точно. Машинне навчання може автоматично проаналізувати мільйони опитувань, коментарів, електронних листів тощо з меншими витратами, часто лише за кілька хвилин. По-друге людина, яка в ручну класифікує данні може допускати помилки при класифікації через відволікання, втому і нудьгу. Машинне навчання застосовує одні й ті самі лінзи та критерії до всіх даних та результатів. Як тільки модель класифікації тексту навчена належним чином, вона працює з неперевершеною точністю.

Розроблені модулі класифікації текстових даних у дипломному проекті вирішують проблему людського фактору та можуть бути широко застосовані у великій кількості сфер.

Перший модуль – це бінарна класифікація текстових даних, а саме визначення емоційного забарвлення. Модель визначає чи рецензія на кінофільм є позитивною або негативною.

Компанії використовують класифікатори настроїв у широкому спектрі додатків, таких як аналіз продукції, моніторинг бренду, маркетингові дослідження, підтримка клієнтів, аналіз робочої сили та багато іншого.

Класифікатор настрою відгуків можна використовувати в онлайн кінотеатрах. По класифікації відгуків створити рейтинг кінофільмів. Категоризація відгуків на кінофільми важлива тим, що вона має значний вплив на процеси мислення споживачів, її можна використовувати не тільки як маркетинговий інструмент для кіностудій, але і передбачати фінансові результати фільму.

Другий модуль – це мультикласова класифікація текстових даних, а саме за текстом новин визначити, до якої з 6 категорій (розваги, бізнес, політика, спорт, техніка) відноситься новина.

Категоризація контенту – це спосіб покращити перегляд або визначити пов'язаний контент на сайті. Такі платформи, як агентства новин, блоги, соціальні мережі тощо, можуть використовувати автоматизовані технології для класифікації та маркування контенту. Також, можна використовувати для персоналізація новинної стрічки до вподобань кожного користувача шляхом визначення категорії новини та пріоритезації її появи.

ВИСНОВКИ

У сучасному світі більшість інформації подається в текстовому форматі. В результаті зростає потреба в автоматизованих системах обробки текстових даних. Для вирішення конкретної задачі класифікації текстів потрібно витратити багато ресурсів, що не є прийнятим особливо при вирішенні складних проблем. Тому необхідно розробити модулі інформаційної систем для вирішення поставленої задачі.

У процесі аналізу предметної області було розглянуто нейронні мережі: типи нейронних мереж, їх робота, проблема та типи класифікації текстових даних. Були проведені огляд та аналіз аналогів, які показали, що в Інтернет просторі не має автоматичних класифікаторів україномовних текстів та це питання мало досліджене. Сформовано постановку задачі. Описано призначення розробки, а також встановлено мету та задачі розробки.

Другий розділ присвячено розробці алгоритму класифікації текстових даних, описані основні етапи класифікації текстів, побудови моделі та оцінки якості моделі.

Заключний розділ роботи містить опис практичної частини. Було проведений вибір програмного забезпечення, детально описано роботу моделі бінарної та мультикласової класифікації текстів, обґрунтовано вибір мережі, її тестування та аналіз отриманих результатів точності навчання.

Отже, цілі, що були поставлені на початки, були досягнені у процесі виконання дипломного проекту. Результатами роботи є розробка модулів для бінарної класифікації текстів відгуків, збору початкових даних з сайту новин та мультикласової класифікації новин за категоріями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Потапова К. Класифікація статей та документів / Катерина Потапова // Матеріали Міжнародної науково-практичної конференції молодих учених, аспірантів та студентів “Інформаційні технології в сучасному світі: дослідження молодих вчених”: тези доповідей, 17 – 18 лютого 2022 р. – Х.: ХНЕУ імені Семена Кузнеця, 2022. – С. 86.
2. Neural Networks [Електронний ресурс]. – 2020. – Режим доступу: https://www.ibm.com/cloud/learn/neural-networks#toc-types-of-n-YgdI1Kt
3. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning [Електронний ресурс]. – 2020. – Режим доступу: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
4. Neural Networks What they are & why they matter [Електронний ресурс] – Режим доступу: https://www.sas.com/en_us/insights/analytics/neural-networks.
5. Jason Brownlee. 4 Types of Classification Tasks in Machine Learning [Електронний ресурс] / Jason Brownlee. – 2020. – Режим доступу: https://machinelearningmastery.com/types-of-classification-in-machine-learning/
6. Free Software for Text Analysis, Text Mining, Text Analytics [Електронний ресурс]. – 2022. – Режим доступу: www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
7. A General Approach to Preprocessing Text Data [Електронний ресурс]. – 2017. – Режим доступу: https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html
8. Data Collection [Електронний ресурс]. – 2021. – Режим доступу: https://www.datarobot.com/wiki/data-collection/

9. Tobias Geisler Mesevage. What Is Data Preprocessing & What Are The Steps Involved? [Електронний ресурс] / Tobias Geisler Mesevage. – 2021. – Режим доступу: https://monkeylearn.com/blog/data-preprocessing/.

10. Data Exploration [Електронний ресурс]. – 2020. – Режим доступу: https://blog.ml.cmu.edu/2020/08/31/2-data-exploration/

11. Model Evaluation [Електронний ресурс] – Режим доступу: https://www.saedsayad.com/model_evaluation.htm

12. Evaluating a machine learning model. [Електронний ресурс]. – 2017. – Режим доступу: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/.

13. Оценка качества в задачах классификации и регрессии [Електронний ресурс]. – 2022. – Режим доступуу: https://neerc.ifmo.ru/wiki/index.php?title=Оценка_качества_в_задачах_классификации_и_регрессии#ROC-.D0.BA.D1.80.D0.B8.D0.B2.D0.B0.D1.8F.

14. Top 5 Word Tokenizers That Every NLP Data Scientist Should Know [Електронний ресурс]. – 2021. – Режим доступу: https://towardsdatascience.com/top-5-word-tokenizers-that-every-nlp-data-scientist-should-know-45cc31f8e8b9

15. Text Normalization for Natural Language Processing (NLP) [Електронний ресурс]. – 2021. – Режим доступу: https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646

16. An introduction to word embeddings for text analysis [Електронний ресурс]. – 2022. – Режим доступу: https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

17. Google Colab : Le guide Ultime [Електронний ресурс]. – 2021. – Режим доступу: https://ledatascientist.com/google-colab-le-guide-ultime/.

18. What Is Python Used For? [Електронний ресурс]. – 2022. – Режим доступу: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

19. Keras : tout savoir sur l’API de Deep Learning [Електронний ресурс]. – 2021. – Режим доступу: https://datascientest.com/keras.

20. What is Tensorflow: Deep Learning Libraries and Program Elements Explained [Електронний ресурс]. – 2022. – Режим доступу: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow#what_is_tensorflow

21. NumPy [Електронний ресурс]. – 2022. – Режим доступу: https://numpy.org/doc/stable/user/absolute_beginners.html.

22. IMDB Dataset of 50K Movie Reviews [Електронний ресурс]. – 2019. – Режим доступу: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews?resource=download&select=IMDB+Dataset.csv

23. Nehal Mohamed Ali Sentiment analysis for movies reviews dataset using deep learning models / Nehal Mohamed Ali, Marwa Mostafa Abd El Hamid and Aliaa Youssif //International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.9, No.2/3, May 2019, Available at SSRN: https://ssrn.com/abstract=3403985

24. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: Sentiment classification using machine learning techniques. In: Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, vol. 10, pp. 79–86. – DOI:10.3115/1118693.1118704

25. Rohith Gandhi. Improving the Performance of a Neural Network. – 2018. – Mode of access: https://towardsdatascience.com/how-to-increase-the-accuracy-of-a-neural-network-9f5d1c6f407d

26. BBC News Train Data | Kaggle [Електронний ресурс]. – 2018. – Режим доступу: https://www.kaggle.com/code/kerneler/starter-bbc-news-train-data-80d7c03e-d/data

ДОДАТКИ

Додаток А

Код програмного модуля бінарної класифікації відгуків

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import numpy as np
df = pd.read_csv('drive/MyDrive/Colab Notebooks/movie_data.csv', engine='python',
quoting = 1, sep=',')
df.head()

from numpy import array
import re
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers.embeddings import Embedding
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from numpy.random import seed
seed(42)
import tensorflow as tf
tf.compat.v1.set_random_seed(42)
docs=df['review']
labels=array(df['sentiment'])
docs[1]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(docs,labels,test_size=0.20)
print(X_train)

all_words = []
for sent in X_train:
  tokenize_word = sent.lower().split(' ')
```

```
  for word in tokenize_word:
    all_words.append(word)
unique_words = set(all_words)
print(len(unique_words))

vocab_size= 340252
X_train=[one_hot(d,vocab_size,filters='!"#$%&()*+,-
./:;<=>?@[\]^_`{|}~',lower=True,split=' ')for d in X_train]
X_test=[one_hot(d,vocab_size,filters='!"#$%&()*+,-
./:;<=>?@[\]^_`{|}~',lower=True,split=' ')for d in X_test]

print(X_train[1])
print(len(X_train[1]))

lens = []
for sent in X_train:
  lens.append(len(sent))
max_length=max(lens)
max_length

max_length=2332
X_train=pad_sequences(X_train,maxlen=max_length,padding='pre')
X_test=pad_sequences(X_test,maxlen=max_length,padding='pre')
print(X_train[1])

from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_acc', mode='max', verbose=1, patience=20,
restore_best_weights=True)

model=Sequential()
model.add(Embedding(vocab_size,32,input_length=max_length))
model.add(Flatten())
model.add(Dense(32,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

print(model.summary())

history=model.fit(X_train, y_train, epochs=20, verbose=1, validation_split=0.4,
batch_size=512, callbacks=[es])
```

```
loss,accuracy=model.evaluate(X_train,y_train,verbose=1)
print('Training accuracy is {}'.format(accuracy*100))

loss,accuracy=model.evaluate(X_test,y_test)
print('Testing accuracy is {}'.format(accuracy*100))

from keras.models import Model
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot4a.png', show_shapes=True,
show_layer_names=True)

import matplotlib.pyplot as plt
def plot_graphs (history,metric):
  plt.plot (history.history[metric])
  plt.plot (history.history['val_'+metric], '')
  plt.xlabel("Epochs")
  plt.ylabel(metric)
  plt.legend([metric, 'val_'+ metric])

plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plot_graphs (history,'acc')
plt.ylim(None,1)
plt.subplot (1,2,2)
plot_graphs (history,'loss')
plt.ylim(0,None)

sample_text=('Saw the movie today and thought it was a good effort, good messages for
kids.')
oh=one_hot(sample_text,vocab_size,filters='!"#$%&()*+,-
./:;<=>?@[\]^_`{|}~',lower=True,split=' ')
print(oh)
pad=pad_sequences([oh],maxlen=max_length,padding='pre')
predictions=model.predict(pad)
print(predictions)

sample_text=('The acting was bad, the dialogs were extremely shallow and insincere.')
oh=one_hot(sample_text,vocab_size,filters='!"#$%&()*+,-
./:;<=>?@[\]^_`{|}~',lower=True,split=' ')
print(oh)
pad=pad_sequences([oh],maxlen=max_length,padding='pre')
predictions=model.predict(pad)
print(predictions)
```

Додаток Б

Код програмного модуля мультикласової класифікації новин

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np

df = pd.read_csv('drive/MyDrive/Colab Notebooks/bbc-text.csv', engine='python',
quoting = 1, sep=',')
df.head()

import tensorflow as tf
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers import Activation
from keras.layers import Embedding
from keras.layers import Bidirectional
from keras.layers.embeddings import Embedding

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

vocab_size = 5000 # make the top list of words (common words)
embedding_dim = 64
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = '<OOV>' # OOV = Out of Vocabulary
training_portion = .8

import csv

articles = []
labels = []
```

```
with open("drive/MyDrive/Colab Notebooks/bbc-text.csv", 'r') as csvfile:
    df = csv.reader(csvfile, delimiter=',')
    next(df)
    for row in df:
        labels.append(row[0])
        article = row[1]
        for word in stop_words:
            token = ' ' + word + ' '
            article = article.replace(token, ' ')
            article = article.replace('  ', ' ')
        articles.append(article)
print(len(labels))
print(len(articles))

labels[:2]

articles[:2]

train_size = int(len(articles) * training_portion)

train_articles = articles[0: train_size]
train_labels = labels[0: train_size]

validation_articles = articles[train_size:]
validation_labels = labels[train_size:]

print("train_size",  train_size)
print(f"train_articles {len(train_articles)}")
print("train_labels", len(train_labels))
print("validation_articles", len(validation_articles))
print("validation_labels", len(validation_labels))

tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_articles)
word_index = tokenizer.word_index

train_sequences = tokenizer.texts_to_sequences(train_articles)

print(train_sequences[10]), len(train_sequences[10])

print(train_sequences[0]), len(train_sequences[0])

train_padded = pad_sequences(train_sequences, maxlen=max_length,
padding=padding_type, truncating=trunc_type)
```

```
train_padded[10]

print("len train_sequnces[0]: ", len(train_sequences[0]))
print("len train_padded[0]: ", len(train_padded[0]))

print("len train_sequences[1]: ", len(train_sequences[1]))
print("len train_padded[1]: ", len(train_padded[1]))

print("len train_sequences[10]: ", len(train_sequences[10]))
print("len train_padded[10]: ", len(train_padded[10]))

validation_sequences = tokenizer.texts_to_sequences(validation_articles)
validation_padded = pad_sequences(validation_sequences, maxlen=max_length,
padding=padding_type, truncating=trunc_type)

print(len(validation_sequences))
print(validation_padded.shape)

print(set(labels))

import numpy as np

label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(labels)

training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))
validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))

#labels = ['entertainment', 'business', 'politics', 'sport', 'tech']
label_tokenizer.word_index

print(training_label_seq[0])
print(training_label_seq[1])
print(training_label_seq[2])
print(training_label_seq.shape)
print('----------')
print(validation_label_seq[0])
print(validation_label_seq[1])
print(validation_label_seq[2])
print(validation_label_seq.shape)

from keras.layers import LSTM
```

```
model = Sequential()

model.add(Embedding(vocab_size, embedding_dim))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(embedding_dim)))
model.add(Dense(6, activation='softmax'))

model.summary()

opt = tf.keras.optimizers.Adam(lr=0.001, decay=1e-6)
model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy'],
)

num_epochs = 10
history = model.fit(train_padded, training_label_seq, epochs=num_epochs,
validation_data=(validation_padded, validation_label_seq), verbose=2)

txt = ["house prices show slight increase prices of homes in the uk rose a seasonally
adjusted 0.5% in february  says the nationwide building society.  the figure means the
annual rate of increase in the uk is down to 10.2%  the lowest rate since june 2001. the
annual rate has halved since august last year  as interest rises have cooled the housing
market. at the same time  the number of mortgage approvals fell in january to a near 10-
year low  official bank of  england figures have shown.  nationwide said that in january
house prices went up by 0.4% on the month and by 12.6% on a year earlier.  we are not
seeing the market collapsing in the way some had feared   said nationwide economist
alex bannister. there have been a number of warnings that the uk housing market may
be heading for a downturn after four years of strong growth to 2004. in november
barclays  which owns former building society the woolwich  forecast an 8%  fall in
property prices in 2005  followed by further declines in 2006 and 2007. and last summer
economists at pricewaterhousecoopers (pwc) warned house  prices were overvalued and
could fall by between 10% and 15% by 2009.  the price of an average uk property now
stands at £152 879. homeowners now expect house prices to rise by 1% over the next
six months  mr bannister said. he said if the growth continued at this level then the bank
of england may increase interest rates from their current 4.75%.   i think the key is what
the bank expects to happen to the housing market. we always thought we would see a
small rise  they thought they would see a small decline.  house prices have risen 0.9%
this year  nationwide said  and if this pace of increase persists  prices would rise by just
under 6% in the year to december. this is slightly above the 0-5% range nationwide
predicts.  further evidence of a slowdown in the housing market emerged from bank of
england lending figures released on tuesday. new mortgage loans  in january fell to 79
```

000 from 82 000 in december the bank said. the past few months have seen approvals fall to levels last seen in 1995. the bank revealed that 48 000 fewer mortgages were approved in january than for the same month in 2004. overall mortgage lending rose by £7.2bn in january marginally up on the £7.1bn rise in december."]

```
seq = tokenizer.texts_to_sequences(txt)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['sport', 'bussiness', 'politics', 'tech', 'entertainment'] #orig

print(pred)
print(np.argmax(pred))
print(labels[np.argmax(pred)-1])
```

txt = ["call to save manufacturing jobs the trades union congress (tuc) is calling on the government to stem job losses in manufacturing firms by reviewing the help it gives companies. the tuc said in its submission before the budget that action is needed because of 105 000 jobs lost from the sector over the last year. it calls for better pensions child care provision and decent wages. the 36-page submission also urges the government to examine support other european countries provide to industry. tuc general secretary brendan barber called for a commitment to policies that will make a real difference to the lives of working people. greater investment in childcare strategies and the people delivering that childcare will increases the options available to working parents he said. a commitment to our public services and manufacturing sector ensures that we can continue to compete on a global level and deliver the frontline services that this country needs. he also called for practical measures to help pensioners especially women who he said are most likely to retire in poverty . the submission also calls for decent wages and training for people working in the manufacturing sector."]

```
seq = tokenizer.texts_to_sequences(txt)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['sport', 'bussiness', 'politics', 'tech', 'entertainment']

print(pred)
print(np.argmax(pred))
print(labels[np.argmax(pred)-1])
```