

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису» До захисту допущено:

УДК 004.008

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Інформаційне забезпечення
робототехнічних систем»**

зі спеціальності 126 «Інформаційні системи та технології»

**на тему: «Система автоматичного аналізу тексту із застосуванням нейронної
мережі»**

Виконав:

студент 2 курсу, групи ІК-21мп
Матвійчук Андрій Юрійович

Керівник:

Доцент кафедри ІСТ, к.т.н., доцент,
Тимошин Юрій Афанасійович

Рецензент:

Доцент кафедри ІПІ, к.т.н., доцент,
Лісовиченко Олег Іванович

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма

«Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Матвійчук Андрій Юрійович

1. Тема дисертації «Система автоматичного аналізу тексту із застосуванням нейронної мережі», науковий керівник дисертації Тимошин Юрій Афанасійович к.т.н., доцент, затверджені наказом по університету від «07» 11 2023 р. № 5168-с
2. Термін подання студентом дисертації «08» 01 2024 р.
3. Об'єкт дослідження - технології автоматичного аналізу тексту.
4. Вихідні дані – методи автоматичного аналізу та обробки текстів, категорії текстів, технології попередньої обробки текстів, типи бібліотечних класифікаторів, характеристики нейронних мереж.
5. Перелік завдань, які потрібно розробити – аналіз наявних систем та їх рішень, попередня розробка та визначення інструментів для створення складної обчислювальної системи, створення алгоритмів, покращення попередньої обробки даних і вилучення важливої інформації з результатів.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу: UML-діаграма класів, архітектура нейронної мережі для аналізу текста, структура компонентів програми, функціональна схема нейронної мережі для аналізу текста, блок-схема алгоритму програми, діаграма послідовності програми, діаграма прецедентів програми, діаграма активності програми.

7. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	04.09.2023	
2	Аналіз існуючих рішень та моделей	06.09.2023	
3	Вибір технології розробки	15.09.2023	
4	Реалізація системи	16.10.2023	
5	Тестування системи та оцінка її якості	28.10.2023	
6	Оформлення роботи відповідно до вимог	12.12.2023	
7	Попередній захист	21.12.2023	
8	Захист	19.01.2024	

Студент

Андрій МАТВІЙЧУК

Науковий керівник

Юрій ТИМОШИН

РЕФЕРАТ

Дисертація складається зі вступу, п`яти розділів та висновку.

Пояснювальна записка містить 103 аркушів тексту, 56 рис., 24 табл., 8 додатків. Перелік використаних джерел складається з 46 найменувань.

Актуальність теми. Сьогодні необхідність автоматичної системи аналізу тексту з використанням нейронних мереж є найбільш перспективною. Ця передова технологія використовує штучний інтелект для автоматичної обробки та розуміння великих обсягів текстових даних. Його використання розповсюджується на бізнес, освіту та соціальні медіа, які надають важливу інформацію на основі відгуків клієнтів, а доступ до цієї інформації становиться швидким, і ця інформація використовується для покращення певних бізнес процесів та соціальних питань.

Мета і задача дослідження. Метою та головним завданням є створення інтелектуальної системи, яка отримує основну інформацію з великої кількості письмового тексту, обробляє її та видає певні результати для подальшого аналізу. Це передбачає створення алгоритмів, які використовують покращення попередньої обробки даних для підвищення точності роботи нейронної мережі, вилучення важливої інформації з результатів та її аналізу.

Об'єкт дослідження. Технології автоматичного аналізу тексту.

Предмет дослідження. Аналіз існуючих методів та алгоритмів, функціональних моделей та обчислювальних систем для розбору та дослідження різноманітного текста, розробка нових алгоритмів для автоматичного аналізу текстових документів.

Методи досліджень. Заради досягнення поставленої мети та задач, було виконано досконалий аналіз методів, характеристик послідовних та непослідовних моделей нейронних мереж, аналіз програмного та технічного забезпечення для реалізації. Сформовані дані для тестового аналізу і проведено тестування.

Наукова новизна. Зключається у здатності виконання автоматичного аналізу та обробки природно мовних текстів з використанням нейронної мережі.

Практичне значення. Головна практична цінність роботи полягає у розробці системи автоматичного аналізу тексту із застосуванням нейронної мережі з метою підвищення ефективності і точності такого аналізу для отримання максимально коректних оцінок та їх подальшого використання в задачах прийняття рішень та аналізу ситуацій в різноманітних сферах роботи.

Ключові слова. ТЕКСТОВИЙ ДОКУМЕНТ, АВТОМАТИЧНИЙ АНАЛІЗ, ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ, СИНТАКСИЧНИЙ АНАЛІЗ CSV, КЛАСИФІКАЦІЯ ТЕКСТУ.

ABSTRACT

The dissertation consists of an introduction, five chapters and a conclusion.

The explanatory note contains 103 sheets of text, 56 figures, 24 tables, and 8 appendices. The list of used sources consists of 46 items.

Actuality of theme. Today, the need for an automatic text analysis system using neural networks is the most promising. This advanced technology uses artificial intelligence to automatically process and understand large volumes of text data. Its use extends to business, education and social media, which provide important information based on customer feedback, and access to this information becomes fast, and this information is used to improve certain business processes and social issues.

The purpose and task of the research. The goal and main task is to create an intelligent system that receives basic information from a large amount of written text, processes it and produces certain results for further analysis. This involves creating algorithms that use improved data pre-processing to increase the accuracy of the neural network, extract important information from the results, and analyze them.

Object of study. Technologies of automatic text analysis.

Subject of study. Analysis of existing methods and algorithms, functional models and computer systems for analysis and research of various texts, development of new algorithms for automatic analysis of text documents.

Research methods. In order to achieve the set goal and tasks, a perfect analysis of methods, characteristics of sequential and non-sequential models of neural networks, analysis of software and technical support for implementation was performed. Generated data for test analysis and conducted testing.

Scientific novelty. It consists in the ability to perform automatic analysis and processing of natural language texts using a neural network.

Practical meaning. The main practical value of the work consists in the development of a system of automatic text analysis using a neural network in order to increase the efficiency and accuracy of such analysis in order to obtain the most correct

estimates and their further use in decision-making tasks and analysis of situations in various areas of work.

Keywords. TEXT DOCUMENT, AUTOMATIC ANALYSIS, DATA PRE-PROCESSING, NEURAL NETWORK LEARNING, CSV ANALYSIS, TEXT CLASSIFICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	10
ВСТУП	11
1 ЗАГАЛЬНИЙ ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ОБРОБКИ ТЕКСТА	13
1.1 Тематичне моделювання	13
1.2 Загальні методи глибинного навчання	14
1.3 Загальні методи Binary Relevance	14
Висновок до розділу 1	15
2 АНАЛІЗ ХАРАКТЕРИСТИК ІСНУЮЧИХ ТИПІВ НЕЙРОННИХ МЕРЕЖ	17
2.1 RNN моделі	17
2.2 LSTM моделі	18
2.3 Моделі трансформерів	20
2.4 BRRN моделі	22
2.5 FNNs модель	24
2.6 SVM модель	25
2.7 Наївний Байєс модель	28
Висновок до розділу 2	30
3 АНАЛІЗ ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ	31
3.1 Python	31
3.2 Jupiter Notebook	33
3.3 Gradio	35
3.4 Бібліотеки	37
3.5 Технічні вимоги	44
Висновки до розділу 3	45
4 РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОГО АНАЛІЗУ ТЕКСТУ	46
4.1 Загальна теорія написання текстової мережі	46
4.2 Початок написання програми	49
4.3 Попередня обробка даних для моделі	51

4.4 Створення послідовної моделі	56
4.5 Прогнозування даних використовуючи модель	60
4.6 Оцінка моделі	61
4.7 Тестування моделі	63
4.8 Аналіз результатів	66
Висновки до розділу 4	77
5 РОЗРОБКА СТАРТАП-ПРОЄКТУ	78
5.1 Опис ідеї	78
5.2 Технологічний аудит ідеї проекту	80
5.3 Аналіз ринкових можливостей запуску стартап-проекту	81
5.4 Розроблення ринкової стратегії проекту	90
Висновок до розділу 5	97
ВИСНОВКИ	98
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	100
ДОДАТОК А	104
ДОДАТОК Б	105
ДОДАТОК В	106
ДОДАТОК Г	107
ДОДАТОК Д	108
ДОДАТОК Е	109
ДОДАТОК Ж	110
ДОДАТОК И	111

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

BR - Бінарна релевантність;
LDA - Латентне виділення Діріхле;
LSA - Прихований семантичний аналіз;
RNN - Рекурентні нейронні мережі;
LSTM - Довга короткочасна пам'ять;
CNN - Сверточні нейронні мережі;
MLP - Багатошарові перцептрони;
SVM - Опорні вектори;
RNN - Рекурентна нейронна мережа;
GRU - Блоки з рекурентними юнітами;
BRNN - Двонаправлена рекурентна нейронна мережа;
ML - Машинне навчання;
FNN - Нейронна мережа прямого зв'язку;
SVM - Опорна векторна машина;
ASM - Світ вибору атрибутів;
KNN - K-Nearest Neighbors;
GMD – Gauss Math Distributions;
TCN - Тимчасові згорткові мережі;
AI – Штучний інтелект.

ВСТУП

На сьогоднішній день, сучасне суспільство, яке дуже сильно пов'язане із новими технологіями стикається із зростаючою кількістю текстових даних, які можна використати для підвищення ефективності розуміння людини або навіть самого контенту даних, але сам процес може потребувати багато ресурсів і часу. Щоб уникнути цього, дослідження автоматичного аналізу тексту за допомогою нейронних мереж стали найважливішою областю дослідження та однією з головною точки відправки .

Ця інноваційна методологія використовує можливості штучного інтелекту для автоматичного отримання важливої інформації з великої кількості письмових або електронних матеріалів. Досліджуючи тонкощі природної мови, це дослідження намагається створити потужну обчислювальну платформу, яка здатна ефективно й точно аналізувати текст та саму його характеристику. Можливі варіанти використання цієї системи різноманітні, починаючи від аналізу певних слів у реченнях та закінчуючи аналізом характеристики цілого тексту, вона забезпечує вирішення проблем, які є дієвими та забезпечують краще розуміння текстів.

Розвиток онлайн-комунікації, взаємодії в соціальних мережах і створення цифрового контенту призвело до потреби аналізу тексту, а саме пости, історії, короткі описи вакансії чи взагалі великі по обсягу тексти чи наукові статті. Створюючи складні алгоритми та моделі, це дослідження має на меті виявити шаблони та важливу інформацію, що міститься в написаному тексті.

Важливість цього дослідження полягає не тільки в теорії ймовірностей, але й у практичних питаннях, які виникають у тих, хто приймає рішення в різних галузях.

У бізнес-розвідці, академічних дослідженнях чи опитуванні громадських настроїв потенційні можливості використання ефективної автоматичної системи аналізу тексту численні й значні. Обговорюються теоретичні основи аналізу

тексту на основі нейронних мереж, їх побудова і що не менш найважливіше це різні підходи із різною ефективністю, а також його можливі практичні застосування, які можуть покращити процеси прийняття рішень у нашому інформаційному світі.

Аналіз текста є основою для багатьох інших технологій, які у свою чергу зараз активно розвиваються і будуть мати велике значення у майбутньому. Тому визначивши чи побудувавши нові підходи та перевіривши їх у справі, люди будуть мати розуміння ефективності даної технології та її підходів.

1 ЗАГАЛЬНИЙ ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ОБРОБКИ ТЕКСТА

1.1 Тематичне моделювання

Тематичне моделювання [1] є важливою складовою аналізу тексту та інформаційної обробки, оскільки воно дозволяє автоматично визначити головні теми та концепти в текстових документах. Два з найбільш відомих методи тематичного моделювання - Latent Dirichlet Allocation (LDA) [2] і Latent Semantic Analysis (LSA) [3] - відкривають можливості для автоматичного визначення тем та призначення їх текстам.

LDA - це ймовірнісна модель, яка припускає, що кожен документ може бути представлений як сума декількох тем. Модель LDA припускає, що кожне слово у документі є результатом вибору теми з розподілу тем. Цей метод дозволяє категоризувати тексти за темами та визначити ймовірності належності кожного слова до певної теми.

LSA, натомість, базується на лінгвістичному аналізі та матричній алгебрі. Він використовує сингулярний розклад матриці термін-документ, щоб визначити семантичну структуру тексту. LSA дозволяє зменшити розмірність даних і виявляти схожість між документами, але він не надає явних тематичних міток.

Binary Relevance [4] може бути використаний для покращення класифікації тексту за темами, структуруючи інформацію на основі тем, визначених LDA або LSA. Цей підхід дозволяє створити багато бінарних класифікаторів, кожен з яких відповідає певній темі. Після цього кожен документ може бути віднесений до одного чи кількох класів (тем), використовуючи ці бінарні класифікатори. Binary Relevance забезпечує гнучкість у визначенні тем і дає можливість покращити точність та надійність класифікації текстового контенту.

Використання Binary Relevance в поєднанні з тематичним моделюванням може зробити аналіз тексту більш точним і корисним у багатьох галузях, включаючи інформаційний пошук, наукові дослідження, аналітику соціальних медіа, та багато інших сфер. Цей підхід відкриває нові можливості для

автоматизованого категоризування та аналізу текстової інформації, роблячи його більш доступним та інформативним.

1.2 Загальні методи глибинного навчання

Глибоке навчання [5] є одним з передових підходів до аналізу та категоризації текстового контенту. Використання нейронних мереж та інших методів глибинного навчання дозволяє створити потужні моделі для класифікації текстів за різними категоріями. В цьому контексті досліджуються сучасні підходи до категоризації тексту за допомогою глибинного навчання та розглядається, як BR може бути використаний для покращення результатів цих методів.

Глибоке навчання використовується для створення нейронних мереж, які можуть аналізувати та класифікувати текстовий контент. Зокрема, рекурентні нейронні мережі (RNN) [6] та зокрема Long Short-Term Memory (LSTM) [7] нейронні мережі дозволяють аналізувати послідовності тексту та враховувати контекст. Крім того, зглиблені нейронні мережі, такі як сверточні нейронні мережі (CNN) [8] і трансформатори [9], можуть аналізувати текст на різних рівнях абстракції, що дозволяє виявляти різні аспекти тексту та визначати категорії.

BR може бути використаний для покращення результатів методів глибинного навчання в контексті категоризації тексту. Застосовуючи BR, можна створити багато бінарних класифікаторів, кожен з яких відповідає певній категорії. Це дозволяє точніше та надійніше визначити, чи текст відповідає певній категорії, і забезпечує більш гнучкий підхід до категоризації.

1.3 Загальні методи Binary Relevance

Binary Relevance є потужним методом у сфері категоризації тексту та аналізу текстового контенту, який дозволяє створювати гнучкі та точні системи класифікації. Цей метод ґрунтується на ідеї використання багатьох бінарних

класифікаторів, кожен з яких відповідає певній категорії або аспекту тексту. Це особливо корисно, коли категорії відрізняються або мають динамічний характер. Його універсальність і гнучкість роблять його набором інструментом у багатьох дисциплінах, включаючи тематичне моделювання, аналіз настроїв і класифікацію.

Застосування BR в контексті класифікації тексту дозволяє досягти високої точності та надійності, оскільки кожен бінарний класифікатор може бути навчений визначати певну категорію. Цей метод особливо корисний у випадках, коли текст може належати до різних категорій або коли категорії є динамічними і змінюються з часом.

Завдяки успішному поєднанню методів аналізу тексту, таких як глибоке навчання та BR, постає можливість створити потужні моделі, які розуміють, класифікують і аналізують текстовий вміст у кількох сценаріях, що призводять до нових способів використання текстової інформації у багатьох сферах людини.

Однією з головних переваг BR є його універсальність і гнучкість. Цей метод може бути успішно використаний в різних сферах, включаючи категоризацію тексту за темами, аналіз емоцій, класифікацію текстового контенту та інше.

Висновок до розділу 1

Моделювання тем дає можливість автоматично розпізнавати основні теми в листових документах за допомогою імовірних підходів, таких як LDA та LSA. Процес класифікації, особливо через бінарну релевантність, створює автоматизовані системи, які позначають категорії текстів. Глибоке навчання використовує нейронні мережі, щоб розрізняти різні рівні складності тексту.

Наприклад, у галузі медицини ці методи можна використовувати для автоматичного аналізу медичних записів та визначення основної причини захворювання або симптомів. У фінансовому секторі їх можна використовувати для визначення переважаючих тенденцій на ринку та прогнозування фінансових

показників корпорацій на основі аналізу звітів. В освітній сфері ці методи можуть сприяти оцінюванню ступеня розуміння студентом матеріалу.

Очевидно, що різні методи обробки тексту сприяють створенню складних та ефективних систем, які можуть вирішувати різноманітні проблеми у різних дисциплінах. Проте дуже важливо розуміти, що конкретний вибір методу часто залежить від конкретного контексту та передбачуваної мети програми.

Поєднання глибокого навчання та тексту може відіграти особливо важливу роль у створенні складних систем, які не тільки мають здатність точно класифікувати текст, а й розуміти його контекст та зв'язки.

2 АНАЛІЗ ХАРАКТЕРИСТИК ІСНУЮЧИХ ТИПІВ НЕЙРОННИХ МЕРЕЖ

2.1 RNN моделі

У сфері обробки NLP [10], RNN є поширеним класом нейронних мереж, які спеціалізуються на повторюваній природі мови. На відміну від традиційних нейронних мереж, RNN призначені для роботи з послідовностями даних, це робить їх ідеальними для вирішення проблем, які вимагають розуміння порядку та залежностей складових частин у послідовності, наприклад, у моделюванні мови, перекладі машин, та аналізу настроїв.

Унікальним атрибутом RNN є їх здатність підтримувати прихований стан, який змінюється з часом, коли мережа обробляє кожен компонент вхідної послідовності.

Цей прихований стан функціонує як пам'ять, яка зберігає інформацію про контекст і залежності попередніх компонентів. Цей повторюваний шаблон полегшує внутрішнє володіння атрибутом, подібним до пам'яті, цей атрибут дозволяє вивчати шаблони, які залежать від порядку, в якому елементи зустрічаються. Приклад вигляду звичайної RNN моделі наведено на рисунку 2.1.

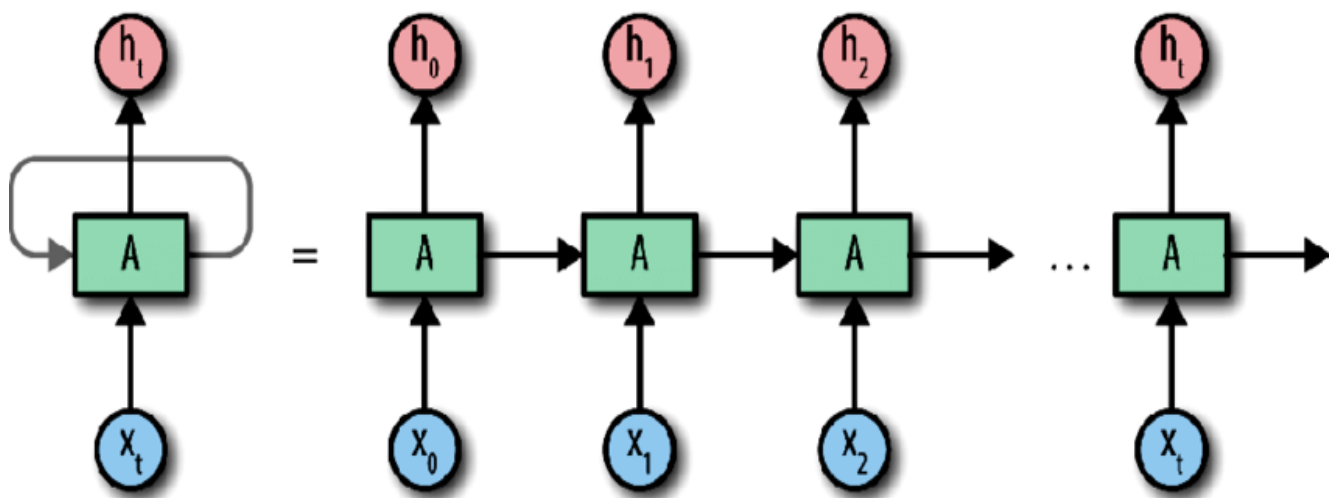


Рисунок 2.1. Приклад вигляду RNN [11]

Однак традиційним RNN важко мати справу з довгостроковими зв'язками в послідовностях. Це називається проблемою зникнення градієнта [12], градієнти, які поширюються в часі, зменшуються з експоненціальною швидкістю, ця проблема перешкоджає здатності мережі розпізнавати віддалені часові залежності.

Щоб уникнути цього недоліку, були розроблені більш просунуті варіанти RNN, включаючи мережі довгострокової короткочасної пам'яті (LSTM) і блоки Gated Recurrent (GRU) [13].

RNN відіграють важливу роль у завданнях розуміння мови, які потребують розуміння часової та лінгвістичної природи мови. Наприклад, у сфері моделювання мови RNN можуть виробляти цікаві та релевантні послідовності слів. Під час аналізу настрою вони можуть визначити настрій речення на основі порядку та розташування слів.

Незважаючи на свою ефективність, RNN мають кілька обмежень, включаючи обчислювальну неефективність і відсутність паралелізму, що призвело до появи конструкцій трансформаторів. Проте RNN все ще служать фундаментальним компонентом NLP, можна продемонструвати постійну еволюцію та вдосконалення структур нейронної мережі, щоб краще справлятися зі складністю послідовних даних.

2.2 LSTM моделі

Мережі довготривалої короткочасної пам'яті, або LSTM, є значним кроком у розвитку нейронних мереж, які, особливо в області NLP мають значний вплив. Розроблені для вирішення проблем, пов'язаних із традиційними рекурентними нейронними мережами, LSTM мають чудову здатність фіксувати та зберігати довгострокові залежності в послідовних даних, що робить їх високоефективними в задачах, які вимагають розуміння контексту.

Фундаментальним компонентом конструкції LSTM є включення комірок пам'яті та складних механізмів стробування. На відміну від RNN, LSTM мають

більш складну пам'ять, яка дає змогу мережі зберігати або втрачати інформацію протягом більш довгих послідовностей. Ця здатність вирішує проблему зникнення градієнта, або труднощі, пов'язані з RNN. Це полегшує збереження важливої інформації на більшій відстані у вхідній послідовності. Приклад роботи LSTM моделі наведено на рисунку 2.2.

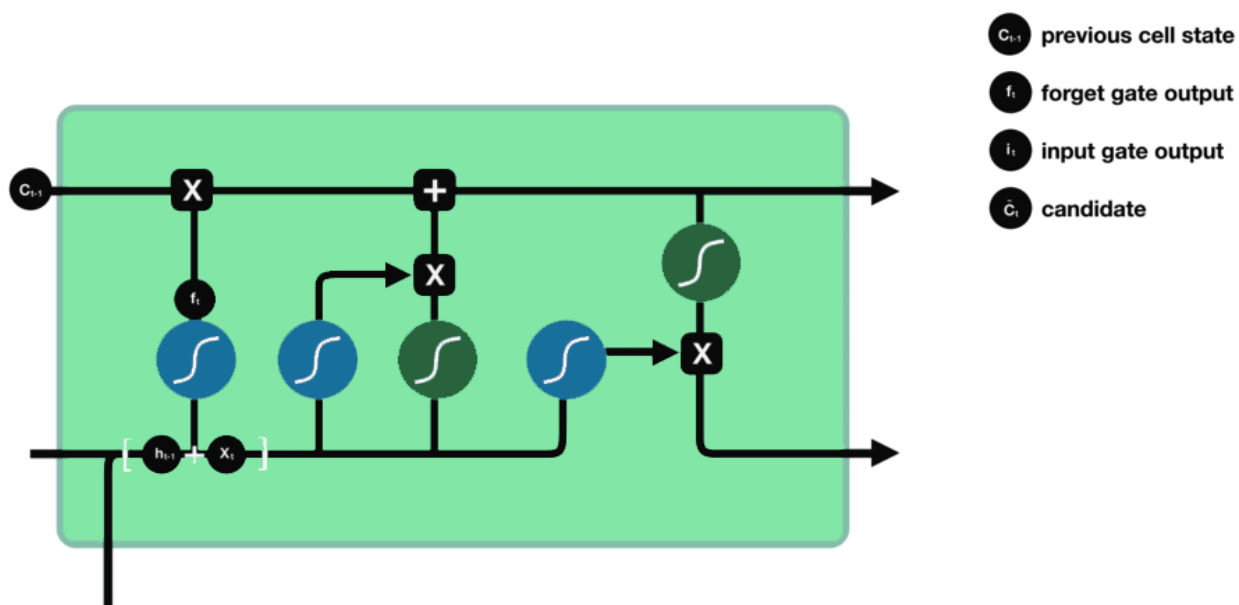


Рисунок 2.2. Приклад вигляду LSTM [14]

Комірка пам'яті в LSTM функціонує як стрічковий конвеєр, і інформація може протікати через різні часові кроки без зменшення інформації. Механізми стробування, включаючи вхідну точку, точку забуття та вихідну точку, контролюють потік інформації в комірці пам'яті, з неї та всередині неї. Цей механізм стробування надає LSTM можливість дізнатися, коли оновлювати комірку пам'яті новою інформацією, коли зберігати наявну інформацію, а коли передавати інформацію на наступний рівень у мережі.

У програмах NLP LSTM продемонстрували свою спроможність у ситуаціях, коли важливі контекст і залежності слів у серії. Наприклад, у сфері мовного моделювання LSTM відіграють важливу роль у створенні зв'язних і контекстуально відповідних послідовностей слів. У машинному перекладі вони

допомагають полегшити переклад речень, зберігаючи складні зв'язки між словами в різних мовах. Крім того, LSTM мають чудову здатність розпізнавати почуття, виражені в реченні в результаті складної взаємодії слів.

Хоча LSTM значно розширили можливості послідовних моделей у NLP, вони все ще мають проблеми. Складність архітектури є значною, і навчання LSTM на великих наборах даних може бути дорогим.

2.3 Моделі трансформерів

Модель трансформатора — це форма глибокого навчання, яку популяризували в 2017 році. Ці моделі стали важливими в обробці природної мови і використовувалися в різноманітних завданнях машинного навчання та штучного інтелекту.

Ці моделі можуть перетворювати текст і мову в режим реального часу. Наприклад, популярні зараз серед туристів додатки можуть спілкуватися з місцевими жителями рідною мовою. Вони сприяють кращому розумінню ДНК і швидкому створенню ліків. Вони мають здатність розпізнавати порушення та запобігати фінансовому шахрайству та шахрайству в сфері безпеки. Подібним чином перетворювачі зору використовуються в програмах комп'ютерного зору. Гарний приклад вигляду роботи трансформатора наведено в рисунку 2.3.

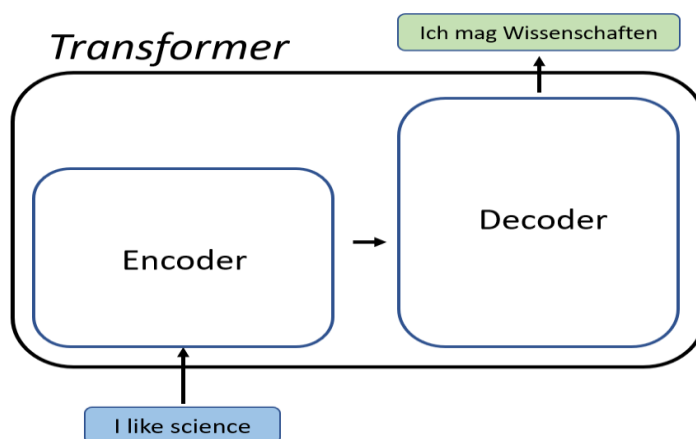


Рисунок 2.3 Приклад спрощеного вигляду трансформатора [15]

Головна інновація моделі трансформатора не залежить від рекурентних нейронних мереж або згорткових нейронних мереж, які мають значні недоліки. Трансформатори приймають вхідні послідовності паралельно, це дуже ефективно для навчання та підтвердження певних висновків. Оскільки не можна просто збільшити кількість графічних процесорів, то необхідно зменшити час обчислення на одиницю. Трансформаторні моделі мають менше часу на навчання, ніж попередні рекурентні моделі нейронних мереж, такі як LSTM.

Моделі трансформаторів використовують серію рівнів, які містять механізми уваги та пересилають нейронні мережі для обробки вхідних даних. Потім ці дані передаються на наступний рівень, який міститиме механізми самоконтролю та нейронні мережі прямого зв'язку.

Яскравий приклад трансформатора є модель від Google яка має назву BERT.

Він викликав переполюх у спільноті машинного навчання через його видатну продуктивність у різноманітних завданнях NLP.

Основною технічною інновацією BERT [16] є використання двонаправленого методу навчання трансформатора, для моделювання мови. Це відрізняється від попередніх зусиль, які зосереджувалися на текстах в цілому.

Результати демонстрації доказують, що мовна модель, яка навчається двонаправлено, краще розуміє мовний контекст і потік, ніж односпрямована мовна модель і описують новий метод навчання під назвою Masked LM (MLM) [17], який дозволяє двонаправлене навчання на моделях, які раніше це було неможливо.

BERT використовує трансформатор, механізм уваги, який вивчає залежності від контексту зв'язки між словами (або підсловами) у тексті. У своїй оригінальній формі трансформатора має дві окремі частини: кодер, який зчитує введений текст, і декодер, який генерує прогноз для завдання.

BERT підходить для різноманітних лінгвістичних завдань, лише незначно збільшуючи основну модель. Приклад вигляду роботи BERT моделі наведено на рисунку 2.4.

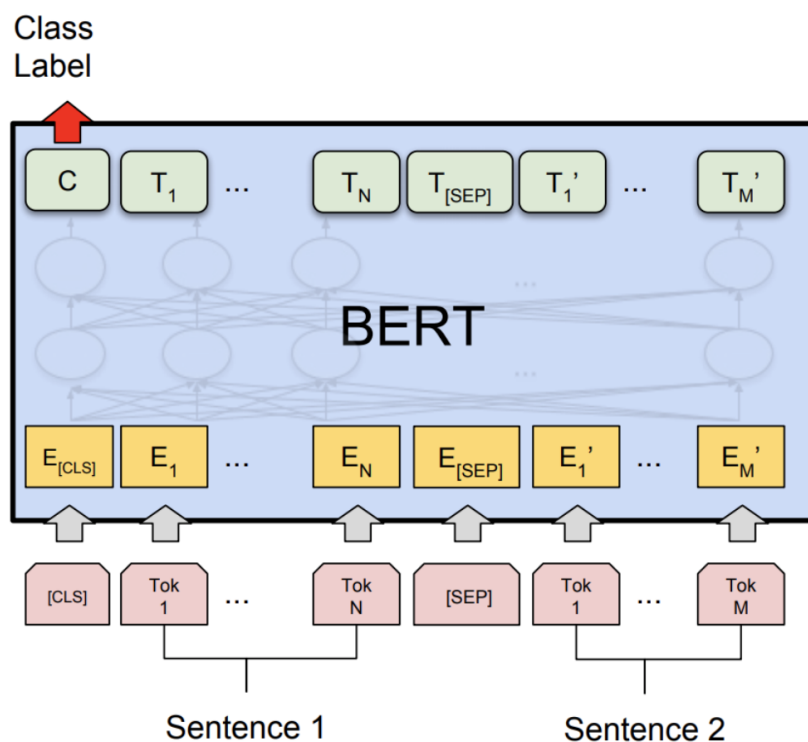


Рисунок 2.4. Приклад вигляду роботи BERT моделі [18]

Оскільки метою BERT є створення мовної моделі, потрібен лише кодер. На відміну від спрямованих моделей, які зчитують текст послідовно (зліва направо або справа наліво), кодер трансформатора зчитує всю послідовність слів одночасно. У результаті його вважають двонаправленим, хоча точніше було б сказати, що він ненаправлений. Цей атрибут дозволяє моделі виводити значення слова на основі навколишніх слів (як ліворуч, так і праворуч від слова).

2.4 BRRN моделі

RNN приймають форму послідовності векторів замість прямої мережі. Ця мережа займається обробкою даних як одного довгого вектора. Кожен вектор обробляється на основі прихованого стану попередньої фази.

Мережа може зберігати інформацію з попередніх кроків у послідовності у вигляді певного типу пам'яті, ця пам'ять обчислюється з урахуванням поточного введення та прихованого стану з попередньої фази. Тому RNN ідеально підходять

для посад, які потребують знання контексту та зв'язків між компонентами послідовності.

Архітектура нейронної мережі, яка називається двонаправленою рекурентною нейронною мережею (BRNN) [19], призначена для обробки послідовних даних. Щоб використовувати інформацію як з минулого, так і з майбутнього контексту у своїх прогнозах, BRNN обробляють вхідні дані як у прямому, так і в зворотному напрямках. Це основна відмінність між BRNN і звичайними рекурентними нейронними мережами. Приклад вигляду BRNN моделі наведено на рисунку 2.5.

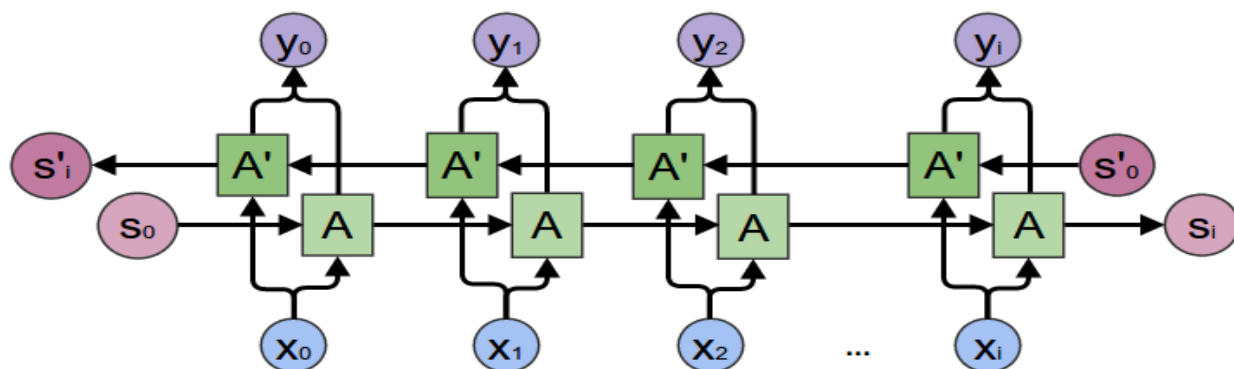


Рисунок 2.5. Приклад вигляду BRNN [20]

BRNN має два окремі повторювані шари, які приховані, один з яких обробляє вхідну послідовність вперед, а інший обробляє її назад. Після цього результати цих прихованих рівнів отримуються та вводяться в останній рівень, який робить прогнози.

У завданнях, які вимагають розуміння всієї послідовності, наприклад, категоризація тексту на кілька класів, двонаправлений характер моделі сприяє кращому розумінню контексту, що підвищує загальну точність і ефективність процесу категоризації.

Порівняно зі звичайними рекурентними нейронними мережами, які є односпрямованими, точність BRNN підвищується, оскільки вона може отримувати інформацію в обох напрямках і мати попередній та майбутній контекст. Оскільки

два додаткові шари можуть як доповнювати один одного, так і надавати кінцевому шару більше даних, використання двох різних прихованих шарів також забезпечує форму регуляризації шару.

Щоб оновити параметри моделі, градієнт розраховується як для прямого, так і для зворотного проходів техніки зворотного поширення в часі, яка зазвичай використовується для навчання BRNN. Вхідна послідовність проходить BRNN за один прохід вперед під час логічного висновку, прогнози робляться на основі комбінованих виходів двох прихованих шарів.

2.5 FNNs модель

Нейронна мережа прямого зв'язку (FNN) [21] — це форма глибокого навчання, яка включає прямий зв'язок. Це як базовий тип нейронної мережі, так і базова архітектура, яка зазвичай використовується для різних завдань машинного навчання, включаючи розпізнавання простих шаблонів, прогнозування майбутніх подій і категоризацію об'єктів.

У FNN (також званому багатокористувачьким персептроном) інформація завжди передається в одному напрямку, ніколи не повертаючись у вихідну позицію. Слова на кшталт «тече в одному напрямку» стосуються інформації, яка передається до нейронів у нейронній мережі. Приклад вигляду роботи FNN моделі наведено на рисунку 2.6.

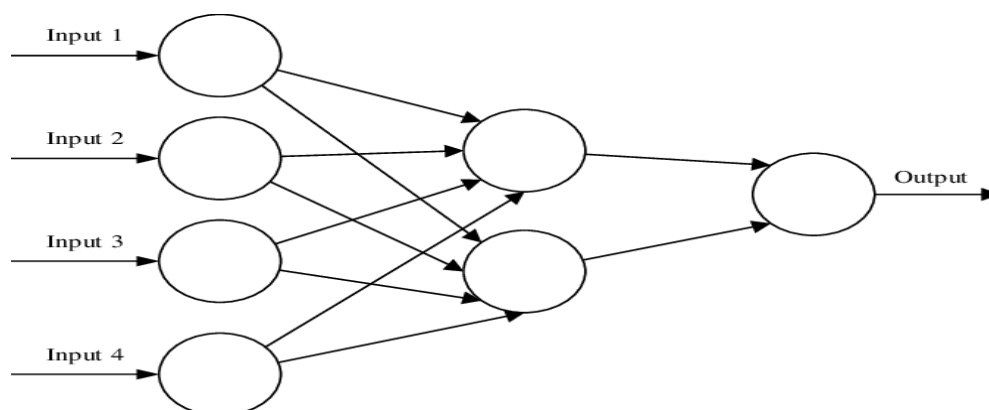


Рисунок 2.6. Приклад вигляду FNN [22]

Вузол — це фундаментальний компонент, який отримує інформацію, щось обчислює на ній і виводить результат за допомогою математичних операцій. Ці нейрони складають шари нейронної мережі, яка може мати кілька шарів, найпростішим з яких є кілька нейронів. У більш складних мережах зазвичай є кілька нейронів.

Як правило, нейронні мережі мають кілька взаємопов'язаних рівнів, включаючи вхідний рівень, приховані шари та вихідний рівень. Однак слід пам'ятати, що склад цих рівнів може відрізнятися залежно від типу мережі чи архітектури або навіть мати інший шаблон (наприклад, RNN).

У нейронній мережі прямого зв'язку, яка зазвичай складається з 1–2 прихованих шарів, ця мережа є більш ефективною для наборів даних, які мають меншу кількість вимірів або ознак і зазвичай менш складні.

2.6 SVM модель

Support Vector Machine (SVM) [23] — це потужний алгоритм машинного навчання, який використовується для класифікації, регресії та виявлення викидів. SVM можна використовувати в різних цілях, включаючи класифікацію тексту, класифікацію зображень, виявлення спаму, розпізнавання рукописного тексту, аналіз експресії текста, виявлення обличчя і виявлення аномальних ознак.

SVM є універсальними та ефективними в різних сценаріях, оскільки він може мати справу з велико вимірними даними та складними зв'язками.

Алгоритми SVM є успішними, оскільки він намагається максимізувати відстань між різними класами в цільовому просторі, щоб диференціювати їх.

Оскільки SVM алгоритм можна розділити на кілька різноманітних моделей, то його загальна ідея роботи буде показана на рисунку 2.7, приклад вигляду лінійної SVM наведено у рисунку 2.8, а нелінійної на рисунку 2.9.

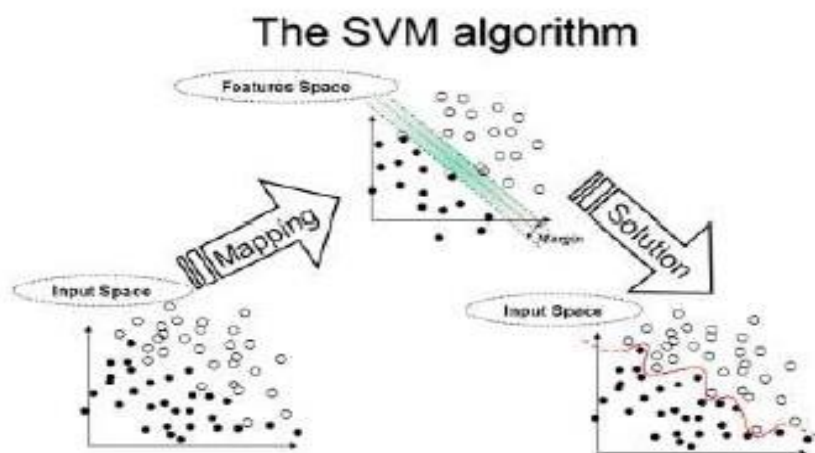


Рисунок 2.7. Приклад вигляду SVM [24]

Support Vector Machine - алгоритм машинного навчання, який контролюється та використовується як для класифікації, так і для регресії. Незважаючи на успішне використання показників регресії, він найбільш ефективний для класифікації. Основна мета алгоритму SVM — знайти оптимальну гіперплощину в N -вимірному просторі, яка може диференціювати точки даних у різних класах у просторі ознак.

Гіперплощина [25] намагається максимально збільшити відстань між найближчими точками різних класів. Довжина гіперплощини залежить від кількості елементів. Якщо кількість вхідних елементів дорівнює двом, то гіперплощина є просто лінією. Якщо кількість вхідних елементів дорівнює 3, то гіперплощина є тривимірною. Важко уявити, скільки функцій можна включити в уявний дизайн, якщо кількість функцій перевищує 3.

Одним з життєздатних варіантів гіперплощини, який максимізує різницю між класами, є гіперплощина, яка представляє найбільше розділення або відмінність між класами.

Природу межі прийняття рішення можна класифікувати як лінійну або нелінійну.

Машини опорних векторів (SVM) поділяються на два основні типи:

- лінійні SVM;
- нелінійний SVM.

Лінійні SVM мають лінійну межу, яка розрізняє точки даних різних класів. Коли дані можна точно розрізнити лінійними засобами, лінійні SVM особливо ефективні. Це означає, що одна пряма (у 2D) або гіперплощина (у вищих вимірах) може повністю диференціювати дані на відповідні класи. Гіперплощина, яка максимізує різницю між класами, є межею для прийняття рішень.

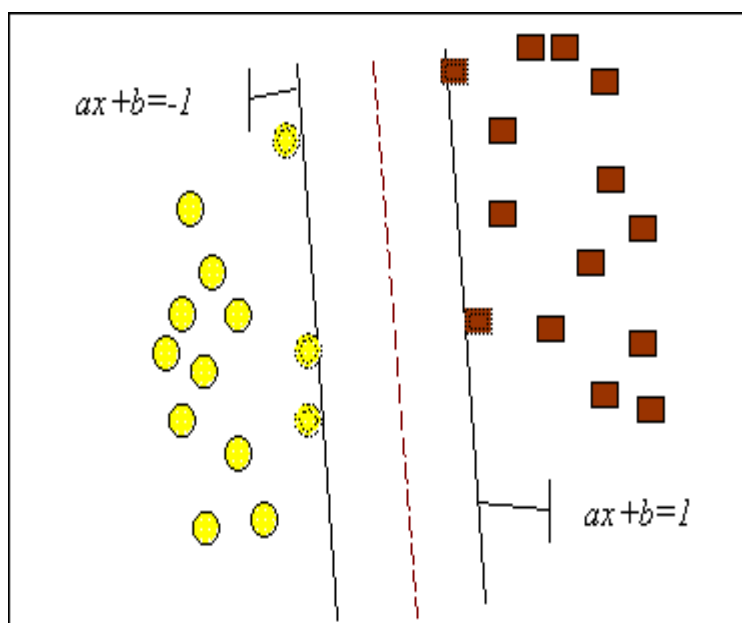


Рисунок 2.8. Приклад вигляду лінійної SVM моделі [26]

Нелінійний SVM використовується для розрізнення між класами, коли для їх розрізнення не можна використовувати пряму лінію (у випадку 2D). Завдяки використанню функцій ядра нелінійні SVM здатні обробляти дані, які за своєю природою нероздільні. Вихідні входні дані змінюються цими функціями на більш вимірний простір, який можна лінійно відрізнити від даних. Лінійний SVM використовується для визначення межі нелінійного рішення в цьому зміненому просторі.

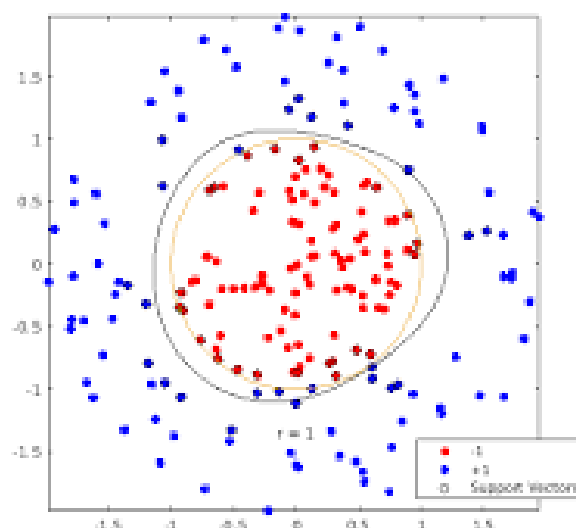


Рисунок 2.9. Приклад вигляду нелінійної SVM моделі [27]

2.7 Наївний Байєс модель

Наївний Байєс (Naive Bayes) [28] — це алгоритм машинного навчання, який використовується для класифікації проблем. Його виведено з теореми ймовірності Байєса. Він в основному використовується для класифікації тексту, який включає великі масиви даних з великими розмірами. Кілька прикладів включають виявлення спаму, сентиментальний аналіз і класифікацію новин.

Це не тільки просто, але й ефективно. За допомогою алгоритму Наївного Байєса легко створювати моделі та робити прогнози. Наївний Байєс – це перший алгоритм, який, як передбачається, буде використано для вирішення проблеми класифікації тексту.

Модель Наївного Байєса – це алгоритм, який вивчає ймовірність речі з певними ознаками, пов'язаними з певним класом або групою. Зрештою, це ймовірнісний розпізнавач. Приклад вигляду роботи Наївного Байєса моделі наведено на рисунку 2.10.

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1				
Whether	No	Yes		
Overcast		4	$=4/14$	0.29
Sunny	2	3	$=5/14$	0.36
Rainy	3	2	$=5/14$	0.36
Total	5	9		
	$=5/14$	$=9/14$		
	0.36	0.64		

Likelihood Table 2				
Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	$0/5=0$	$4/9=0.44$
Sunny	2	3	$2/5=0.4$	$3/9=0.33$
Rainy	3	2	$3/5=0.6$	$2/9=0.22$
Total	5	9		

Рисунок 2.10. Приклад вигляду роботи Наївного Байєса [29]

Наївний алгоритм Байєса вважається «наївним», оскільки він вважає, що наявність певної функції не пов'язана з наявністю інших ознак.

Наприклад, якщо спробувати ідентифікувати фрукт за його кольором, формою та смаком, то фрукт помаранчевого кольору, сферичної форми та коричневого кольору, швидше за все, буде помаранчевим.

Навіть якщо ці властивості залежать одна від одної або наявність інших властивостей, усі ці властивості окремо відповідають за ймовірність того, що цей фрукт буде помаранчевим, тому його називають «наївним».

Частина «Байєса» стосується статистики та філософа Томаса Байєса, а також теореми, названої на його честь, теореми Байєса, яка є основою для наївного алгоритму Байєса.

Як це функціонує? Припустимо, що новий елемент буде класифікований як найбільш ймовірний клас умовної ймовірності, описаний вище.

Висновок до розділу 2

Існують дуже багато різноманітних моделей, кожна з яких має свої певні переваги або недоліки.

Послідовні моделі спеціально створені для тимчасових даних, вони зосереджені на порядку обробки вхідних компонентів, що ідеально підходить для програм, таких як обробка природної мови. Приклади нейронних мереж, які є рекурентними, включають рекурентні нейронні мережі (RNN), мережі довготривалої короткочасної пам'яті (LSTM) і трансформатори.

Непослідовні моделі, такі як нейронні мережі прямого зв'язку (FNN) і опорні векторні машини (SVM), припускають, що вхідні компоненти незалежні один від одного, що ідеально підходить для таких класів, як класифікація зображень і прийняття рішень. У класифікації тексту, яка включає три або більше класів, вибір моделі базується на вимогах завдання.

Підсумовуючи, не можна сказати, що певна модель має велику перевагу над іншими чи навпаки. Тут необхідно дивитися на різні сценарії під різним кутом і пробувати застосовувати різні моделі разом для досягнення кращого результату, оскільки доведено, що деякі моделі пішли від інших або краще працюють в парі із іншими.

3 АНАЛІЗ ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Python

Python [30] — це інтерпретована мова програмування високого рівня, яку легко читати та розуміти, ця мова також надзвичайно зручна та універсальна. З моменту створення вона стала більш корисним для розробників, спеціалістів із обробки даних, дослідників та інших.

Оскільки Python є універсальною мовою, яку можна використовувати в різноманітних проектах, її проста природа ідеально підходить для автоматизації завдань, створення веб-сайтів або програмного забезпечення й особливу аналізу даних.

Завдяки своєму стислому та зручному для читання синтаксису Python став улюбленою мовою для веб-розробки та розробки програмного забезпечення. Це дозволяє розробникам легко створювати складні багатопроTOCOLьні програми. Фактично, багато з найбільш широко використовуваних програм були створені з використанням Python.

Крім того, спільнота Python з відкритим вихідним кодом пропонує розробникам безліч багаторазового коду, фреймворків і допомоги. Яскравим прикладом цього є Django [31], фреймворк Python, створений досвідченими розробниками для прискорення розробки додатків і обходу потенційних перешкод.

Однією з головних переваг Python є його здатність автоматизувати звичайні процеси, керовані людиною. З Python можливо навчитися автоматизувати більшість речей, використовуючи попередньо написаний код або вбудовані модулі. Крім того, мається можливість створювати спеціальні дії, які мають певні форми.

Подібно до того, як Python може допомогти науковцям у роботі з великими наборами даних, Python також використовується у фінансовому секторі для швидкого виконання складних обчислень.

Фондові ринки створюють великі обсяги даних, і Python можна використовувати для імпорту або підрахунку даних про ціни акцій і створення

стратегій на основі алгоритмів, які визначають потенційні торгові можливості. Цю мову також можна використовувати для оптимізації портфелів, зниження ризиків, моделювання фінансів і візуалізації криптовалют.

Крім того, Python присутній у деяких із найскладніших технологій штучного інтелекту (AI) [32]. Насправді це одна з найпопулярніших мов для ШІ. Лаконічний і зрозумілий код Python полегшує розробку послідовних потужних систем, а його велика бібліотека забезпечує численні інфраструктури.

Python є найпопулярнішою мовою програмування в галузі науки про дані. Оскільки його синтаксис простий для розуміння та може бути змінений, люди з обмеженим досвідом розвитку можуть легко вивчити Python і використовувати його для маніпулювання даними для досліджень, звітності, прогнозованого чи регресійного аналізу тощо. Збір і обробка даних може бути повільним процесом для спеціалістів із обробки даних.

Крім того, Python є однією з найпопулярніших мов, яка використовується для навчання моделей машинного навчання (ML). Завдяки спеціальним методам ці моделі можуть аналізувати та розпізнавати закономірності в даних, що дозволяє їм робити прогнози або приймати рішення на основі даних. Рішення також постійно адаптуються на основі результатів попередніх наборів даних для вирішення нових ситуацій.

Переваги використання Python наступні:

- простота, код Python простий, і його можуть читати нові розробники, де це корисно для проектів машинного та глибокого навчання. Завдяки простому синтаксису створення програм, Python є простим порівняно з іншими мовами програмування. Крім того, це полегшує тестування алгоритмів без фактичної їх реалізації;

- гнучкі комбінації, проекти Python можна комбінувати з іншими програмними додатками, написаними різними мовами. Це означає, що набагато простіше поєднати його з іншими автоматизованими проектами, які написані різними мовами. В додаток, оскільки Python є портативним і розширюваним, він

здатний виконувати лінгвістичні завдання різними мовами. Універсальність Python корисна для науковців із обробки даних і розробників, які хочуть використовувати машинне навчання для навчання моделей;

- чудова спільнота, схожа на спільноту JavaScript [33] або .NET [34], також існує спільнота на основі Python і що саме найголовніше, ця спільнота далеко не маленька, а навпаки – дуже велика. Кожен може отримати доступ до всього, що йому потрібно, включно з розробкою.

Це найважливіші переваги Python для машинного навчання. Однак є ще багато, на які також варто звернути увагу. Загалом, Python є широко поширеним та дружною мовою для будь-кого через велику кількість переваг.

3.2 Jupiter Notebook

Jupyter Notebook [35] відомий як потужний, універсальний інструмент для науковців, дослідників і викладачів. Він надає ряд переваг, які сприяють його широкому застосуванню:

Jupyter Notebook — це комп'ютерне середовище, яке дозволяє користувачам писати та виконувати код безпосередньо. Цей підхід є особливо вигідним для дослідження, аналізу та візуалізації даних. Оскільки користувачі можуть одразу оцінити вплив свого коду на дані, цей підхід є вигідним.

Однією з найважливіших переваг Jupyter Notebook є його підтримка кількох мов програмування, включаючи, але не обмежуючись ними, Python, R і Julia. Ця гнучкість дозволяє користувачам вибирати мову, яка найбільше відповідає їхнім потребам, на одному комп'ютері, що полегшує співпрацю між дисциплінами та різними методами аналізу.

Jupyter Notebook полегшує додавання мультимедійних компонентів, включаючи зображення, графіки та інтерактивні візуалізації. Ця здатність має вирішальне значення для створення інформативних і візуально приємних звітів, вона ідеальна для вчених, які хочуть ефективно повідомити про свої висновки.

Також полегшують інтеграцію Markdown, коду та документації, дозволяючи користувачам мати повне розуміння продукту без кодування. Таке поєднання коду та тексту в одному документі підвищує ясність аналізу, це чудовий метод для створення комплексних та добре задокументованих звітів.

Використовуючи величезні ресурси Python, Jupyter Notebooks може використовувати численні бібліотеки та інструменти для обробки даних, машинного навчання та наукових обчислень. Цей зв'язок із такими популярними бібліотеками, як NumPy, Pandas, Matplotlib і scikit-learn, має значний вплив на можливості Jupyter для різноманітних завдань, керованих даними.

Jupyter Notebook легко запускати в хмарних середовищах, це дозволяє користувачам користуватися перевагами масштабованих обчислювальних ресурсів і співпрацювати над проектами, не турбуючись про обмеження своїх локальних комп'ютерів. Хмарні платформи, такі як Google Colab [36] і Microsoft Azure Notebooks [37], мають плавну інтеграцію з Jupyter, ця платформа надає додаткові функції та доступність.

Також ця технологія зазвичай використовуються в навчальних закладах для навчання програмуванню, науковим дослідженням даних і науковим ідеям. Інтерактивний характер зошитів полегшує експерименти з кодом, візуалізаціями та складними концепціями, усі з яких призначені для покращення досвіду навчання.

Jupyter — це безкоштовний проект, який має велику спільноту, яка бере участь у його розвитку. Цей спільний підхід сприяє постійному розвитку, регулярним оновленням і великій кількості ресурсів і розширень, доступних через платформу Jupyter.

Загалом переваги Jupyter Notebook включають його інтерактивне обчислювальне середовище, підтримку кількох мов, мультимедійні компоненти, легкість доступу та відтворюваність, документовану інформацію, розгалужену систему та сумісність із хмарою.

3.3 Gradio

Коли справа доходить до включення нейронних мереж у програми, Gradio [38] виділяється як привабливий варіант. Він служить плавним зв'язком між моделями машинного навчання та інтерфейсами кінцевого користувача, що робить його популярним вибором.

Що відрізняє Gradio від інших, так це його зручні контракти(методи), що дозволяє розробникам легко проектувати інтерактивні інтерфейси для моделей нейронних мереж, навіть не маючи глибоких знань у розробці інтерфейсу.

Інтегруючи Gradio у проекти із залученням нейронних мереж, можна ефективно підвищити доступність програм машинного навчання. Це дозволяє розробникам швидко розгортати та поширювати моделі серед різних зацікавлених сторін, надаючи тим самим ширшому колу осіб можливість взаємодіяти з нейронними мережами та розуміти їх потенціал.

Така доступність відповідає головній меті сприяння інклюзивності та зрозумілості у сфері машинного навчання навіть для тих, хто не має спеціальних знань. Пригляд вигляду даного інтерфейса наведено на рисунку 3.1.

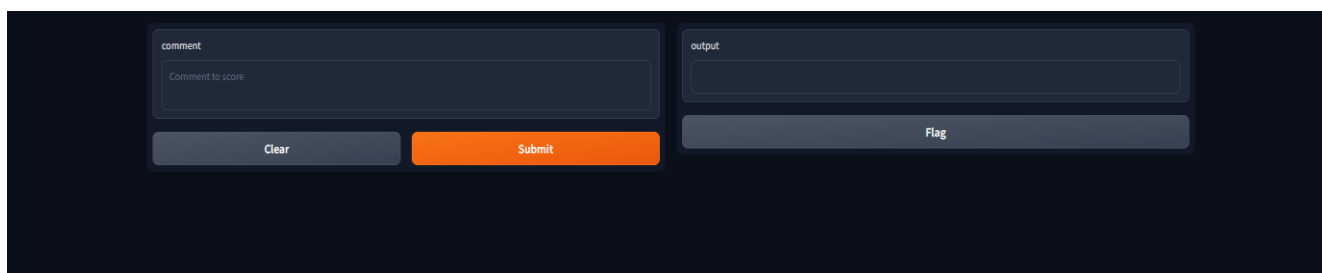


Рис 3.1 Приклад вигляду gradio інтерфейса у web

Процес розробки інтерфейсу для моделей нейронних мереж різко прискорюється за допомогою Gradio, що призводить до швидкого створення прототипів. За лічені хвилини розробники можуть створювати інтерактивні інтерфейси, тим самим прискорюючи цикл розробки та сприяючи швидкій ітерації та експериментуванню.

Процес створення зручного інтерфейсу Gradio спрощує розгортання моделей машинного навчання в додатках реального світу, ефективно мінімізуючи час і зусилля, які зазвичай пов'язані з розробкою інтерфейсу.

Gradio пропонує значну перевагу у своїй здатності вміщувати широкий набір форматів введення, включаючи текст, зображення та аудіо. Ця гнучкість гарантує, що Gradio можна ефективно використовувати в широкому спектрі додатків нейронної мережі, починаючи від обробки природної мови до комп'ютерного зору та аналізу звуку.

Така універсальність позиціонує Gradio як безцінний ресурс для розробників, які займаються різноманітними машинними навчаннями.

Функція зворотного зв'язку в режимі реального часу, яку надає Gradio, виявилася надзвичайно корисною для проєктів із застосуванням нейронних мереж. Ретельно відстежуючи взаємодію користувачів із моделлю, розробники можуть отримати цінну інформацію про її продуктивність і визначити області, які потребують вдосконалення.

Цей ітеративний цикл зворотного зв'язку значно покращує процес розробки моделі, дозволяючи безперервно вдосконалювати її на основі інформації та досвіду користувача.

Крім того, Gradio пропонує інтеграцію з широко використовуваними фреймворками глибокого навчання, такими як TensorFlow [39] і PyTorch [40], що забезпечує плавну сумісність з різними архітектурами нейронних мереж. Ця адаптивність дає змогу розробникам без особливих зусиль інтегрувати свої існуючі моделі в структуру Gradio, усуваючи потребу в значних налаштуваннях моделі.

Підводячи підсумок, Gradio відіграє вирішальну роль у забезпеченні інтеграції нейронних мереж у програми, пропонуючи процес створення зручного інтерфейсу. Він також демократизує доступ до машинного навчання, підтримує швидке створення прототипів, вміщує різні типи введення, забезпечує зворотний

зв'язок у реальному часі та легко інтегрується з популярними фреймворками глибокого навчання.

Ці функції разом роблять Gradio безцінним ресурсом для розробників, які прагнуть покращити доступність і зручність використання своїх моделей нейронних мереж у реальних сценаріях.

3.4 Бібліотеки

Під капотом проєкта будуть використовуватися багато різноманітних бібліотек, кожна з яких має велике значення при праці із нейроном мережею та є необхідним компонентом для успішного створення моделі.

Можливо отримати застосований список бібліотек за допомогою команди, приклад якої наведено в рисунку 3.2.

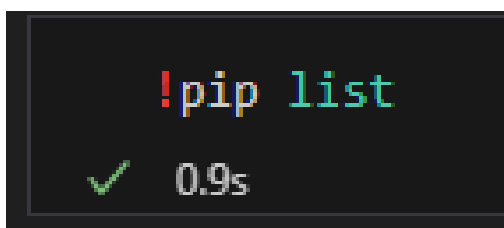
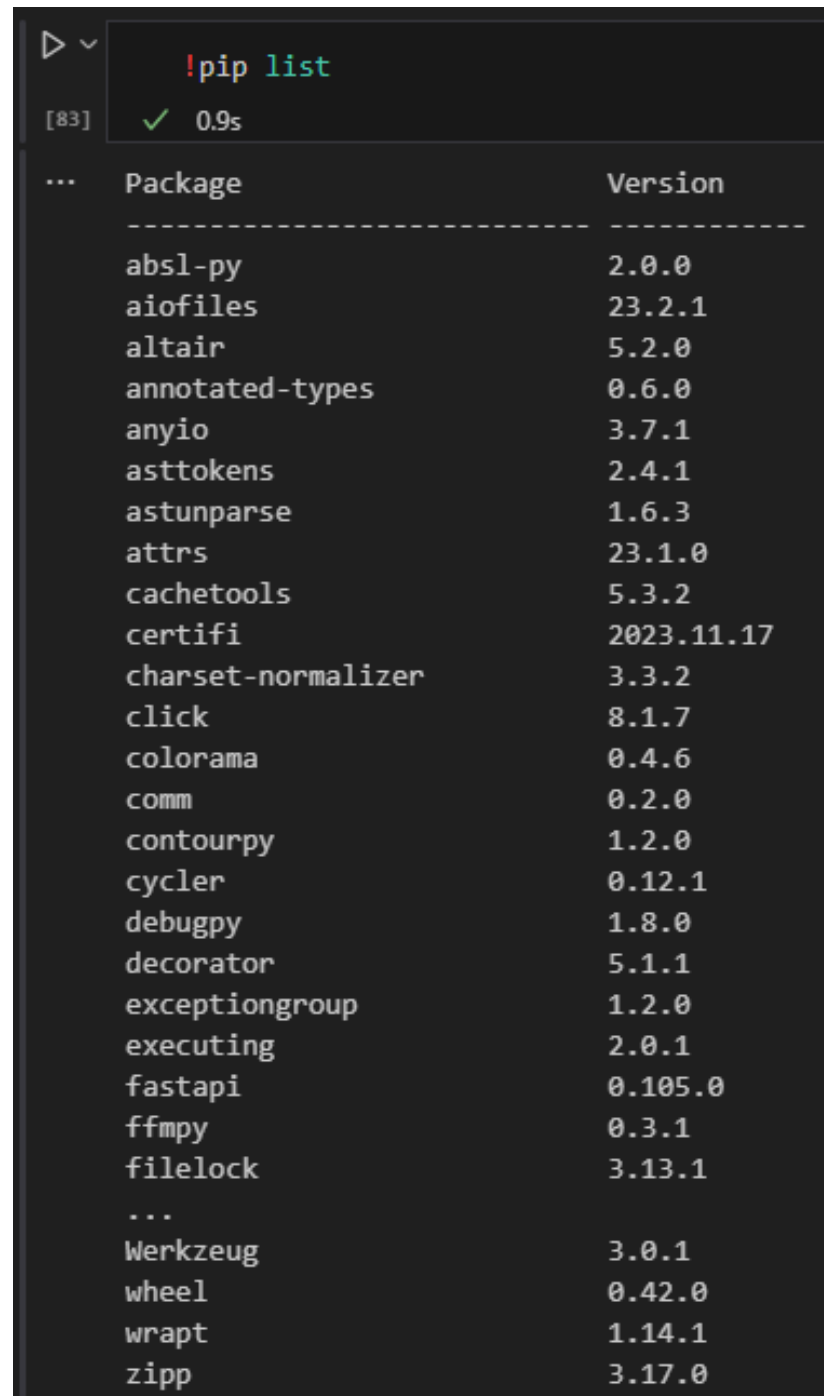


Рис 3.2 Команда для отримання необхідних бібліотек

Після цього буде надано список усіх встановлених на даний момент бібліотек в проєкті.

Якщо кількість буде невелика то можна буде спостерігати його повний список, але скоріше за все кількість бібліотек буде дуже великою, тому буде надано вигляд скорочених бібліотек, приклад якого наведено на рисунку 3.3.



```
!pip list
[83] ✓ 0.9s
```

Package	Version
absl-py	2.0.0
aiofiles	23.2.1
altair	5.2.0
annotated-types	0.6.0
anyio	3.7.1
asttokens	2.4.1
astunparse	1.6.3
attrs	23.1.0
cachetools	5.3.2
certifi	2023.11.17
charset-normalizer	3.3.2
click	8.1.7
colorama	0.4.6
comm	0.2.0
contourpy	1.2.0
cyclor	0.12.1
debugpy	1.8.0
decorator	5.1.1
exceptiongroup	1.2.0
executing	2.0.1
fastapi	0.105.0
ffmpeg	0.3.1
filelock	3.13.1
...	
Werkzeug	3.0.1
wheel	0.42.0
wrapt	1.14.1
zipp	3.17.0

Рис 3.3 Список необхідних бібліотек

Необхідно більш детальніше роздивитися бібліотеки які мають велику цінність для моделі. Перелік використаних засобів розробки наведено у таблиці 3.1

Таблиця 3.1 – Перелік використаних засобів розробки

Бібліотека/фреймворк	Опис
tensorflow	TensorFlow, розроблений командою Google Brain, є системою машинного навчання з відкритим кодом. Його основна мета — сприяти створенню та навчанню моделей машинного навчання, зокрема глибоких нейронних мереж, віддаючи пріоритет масштабованості та адаптивності. TensorFlow знаходить широке застосування в ряді завдань, таких як розпізнавання зображень і мови, обробка природної мови та числові обчислення.
zipper	Zipper — це бібліотека Python, яка надає функції, пов'язані з роботою з ZIP-файлами. Це зменшує складність читання та запису ZIP-файлів, що вигідно розробникам, які хочуть використовувати стислі дані у своїх проектах Python. Бібліотека має простий інтерфейс, який полегшує роботу з ZIP-файлами та створення або розпакування стислих архівів.
tensorboard	TensorBoard — це веб-інструмент візуалізації, який постачається з TensorFlow, популярною системою машинного навчання з відкритим

Бібліотека/фреймворк	Опис
	кодом. Це дозволяє користувачам візуалізувати та контролювати різні аспекти процесу навчання моделі машинного навчання,. Показники навчання та перевірки, графіки моделі та вбудовування. TensorBoard допомагає дослідникам і розробникам отримати уявлення про продуктивність і поведінку моделей TensorFlow, щоб краще зрозуміти, налагодити та оптимізувати свої моделі.
scipy	SciPy — це бібліотека з відкритим кодом на Python для наукових та інженерних обчислень, оскільки це створено на основі NumPy і надає додаткові функції для оптимізації, обробки сигналів і зображень, статистичного аналізу, лінійної алгебри, інтеграції, інтерполяції тощо, то це надає можливість широко використовується в наукових дослідженнях, розробці та аналізі даних завдяки широкому спектру інструментів і алгоритмів.
scikit-learn	Scikit-learn — популярна бібліотека машинного навчання для Python, яка надає прості й ефективні інструменти для аналізу даних і моделювання. Це

Бібліотека/фреймворк	Опис
	охоплює різноманітні методи машинного навчання для класифікації, регресії, кластеризації, зменшення розмірності тощо. Scikit-learn пов'язаний із своїм відповідним і послідовним API, що робить його зручним як для початківців, так і для досвідчених тих, хто вивчає машинне навчання. Ця технологія зазвичай використовується в наукових колах і промисловості для таких цілей, як попередня обробка даних, оцінка моделей і створення конвеєрів машинного навчання.
os	os — це превентивний багатозадачний модуль у Python, який полегшує взаємодію з операційною системою. Він надає численні функції для проектів, такі як керування файлами та каталогами, робота зі шляхами, змінними середовища тощо. Модуль os не залежить від платформи, що дозволяє Python безперебійно працювати в різних операційних системах. Розробники використовують його для виконання завдань, пов'язаних із системою, і взаємодії з основною

Бібліотека/фреймворк	Опис
	файловою системою та середовищем у своєму коді Python.
pandas	Pandas — це потужна автономна платформа для обробки та аналізу даних для Python. Це полегшує створення структур даних, таких як DataFrames, які вправно працюють із попередньо структурованими даними, такими як дані, що зберігаються в таблицях у файлах CSV або SQL. Pandas зменшує складність таких завдань, як очищення, дослідження, перетворення та аналіз даних, тому його зазвичай використовують у процесах аналізу та науки про дані.
keras	Keras — популярний API нейронної мережі для Python, який зазвичай використовується для створення та навчання складних моделей глибокого навчання. Він простий у використанні та може бути реалізований поверх TensorFlow, Theano або CNTK. Keras зазвичай використовується для таких програм, як розпізнавання зображень і розуміння природної мови.
matplotlib	Matplotlib — це популярна бібліотека Python, яка полегшує створення статичних, анімованих та

Бібліотека/фреймворк	Опис
	інтерактивних візуалізацій у різних формах. Він пропонує універсальний і повний набір інструментів для побудови графіків, які можна використовувати для створення різноманітних графіків і діаграм, які підходять для аналізу даних і наукової візуалізації. Matplotlib часто поєднується з іншими бібліотеками, такими як NumPy і Pandas, щоб проводити аналіз даних і маніпуляції.
urllib	urllib — це бібліотека Python, яка містить різноманітні інструменти, пов'язані з обробкою URL-адрес. Він містить такі компоненти, як urllib.request, який використовується для відкриття та читання URL-адрес, urllib.parse, який використовується для аналізу URL-адрес, і urllib.error, який використовується для вирішення винятків, пов'язаних з операціями URL-адрес. urllib часто використовується для проектів, які отримують дані з Інтернету, запитують HTTP-сервіси та використовують URL-адреси в коді Python.
typing	Модуль набору тексту є компонентом стандартної бібліотеки Python, який

Бібліотека/фреймворк	Опис
	допомагає ідентифікувати тип. Це полегшує додавання інформації про типи до параметрів функцій, значення, що повертається функціями, і атрибути класів. Ця інформація призначена, щоб допомогти розробникам розпізнавати очікувані типи змінних і, таким чином, покращити читабельність їх коду. Хоча це не впливає на поведінку коду під час виконання, підказки щодо типів можуть використовуватися інструментами статичного аналізу та IDE, щоб розпізнавати потенційні помилки, пов'язані з типами, і покращувати інтелектуальність коду. Модуль набору тексту містить такі класи та функції, як List, Tuple, Dict, Union і TypeVar, призначені для надання допомоги у вираженні більш складної інформації про тип.

3.5 Технічні вимоги

Оскільки нейронна мережа потребує певних ресурсів для роботи, як і сам python і враховуючи, що не в кожного юзера може матися графічна карта, для того щоб все це працювало, маємо наступний набір технічних вимог:

- оперативна пам'ять – 4 ГБ або більше;

- жорсткий диск – 4ГБ або більше;
- встановлений компілятор мови Python;
- встановлений Pip [41];
- встановлені найнеобхідніші бібліотеки з таблиці 3.1.

Висновки до розділу 3

Були розглянуті різноманітні інструменти для написання нейронних мереж та розглянуті найнеобхідніший пакет мінімальних бібліотек для запуску проєкта. Також було проаналізовано мінімальні технічні вимоги щодо комп'ютера користувача.

Серед розглянутих інструментів було розглянуто популярну мову високого рівня Python, яка дуже добре зарекомендувала себе серед різноманітних розробників по всьому світові завдяки своїм здібностям розробки веб-сайтів, обробку даних, автоматизацію та розробку систем штучного інтелекту.

Було зроблено технічний аналіз необхідних бібліотек для праці із тестовою нейронною мережею та детально розглянуті специфіку їх роботи та застосування. Було виявлено та доказано важливість таких бібліотек як os, scikit-learn, tensorflow, keras й інші. Оцінені технічні вимоги.

4 РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОГО АНАЛІЗУ ТЕКСТУ

4.1 Загальна теорія написання текстової мережі

В першу чергу необхідно датасет із даними, для того щоб в подальшому передати в текстову нейронну мережу для її подальшої роботи.

Загальний вигляд та формат цього файла буде в CSV або Excel формат, тобто таблиця яка вміщувати необхідну інформація в певних колонках.

Для вирішення проблеми, в першу чергу необхідно зрозуміти загальну структуру датасета, який буде використовуватися для навчання нейронної моделі. Приклад полів які будуть застосовані наведено на рисунку 4.1.

	A	B	C	D	E	F	G
1	textID	text	selected_text	sentiment	negative	neutral	positive

Рис 4.1 Загальна структура нашого датасета

Присутні наступні текстові поля із значеннями:

- textID – унікальний ідентифікатор тексту. Можна використати для розуміння унікальності даних а також для розуміння проблемних даних із якими можливо зустрітись в подальшому;
- text – сам текст який буде використовуватися. Складається з різних слів. Може бути як пустим так і заповнений різними словосполученнями;
- selected_text – обрана частина тексту;
- sentiment – загальний стан тексту. 0 – нейтральне, 1 – негативне, 2 позитивне;
- negative/neutral/postivie – двоїчний стан який вираховується за допомогою формули. Дивиться на стан sentiment і якщо відповідає стану то ставить 1, якщо ні – 0.

Сам датасет був взятий з платформи Kaggle [42], але був модифікований для поставлених задач, а саме перероблене значення полів sentiment замінивши слова на цифри і впроваджені додаткові поля для кращої ідентифікації тексту.

Також були додані додаткові дані, та перероблені певні інші, які можна побачити на рисунку 4.2.

1	textID	text	selected_text	sentiment	negative	neutral	positive
2	cb774db0d1	I'd have responded, if I were goir	I'd have responded, if I were going	0	0	1	0
3	549e992a42	Sooo SAD I will miss you here in S	Sooo SAD	1	1	0	0
4	088c60f138	my boss is bullying me...	bullying me	1	1	0	0
5	9642c003ef	what interview! leave me alone	leave me alone	1	1	0	0
6	358bd9e861	Sons of ****, why couldn't they p	Sons of ****,	1	1	0	0
7	28b57f3990	http://www.dothebouncy.com/si	http://www.dothebouncy.com/smf - some shameless plug	0	0	1	0
8	6e0c6d75b1	2am feedings for the baby are fun	fun	2	0	0	1
9	50e14c0bb8	Soooo high	Soooo high	0	0	1	0
10	e050245fbd	Both of you	Both of you	0	0	1	0
11	fc2cbefa9d	Journey!? Wow... u just became c	Wow... u just became cooler.	2	0	0	1
12	2339a9b08b	as much as i love to be hopeful, i	as much as i love to be hopeful, i reckon the chances are mi	0	0	1	0
13	16fab9f95b	I really really like the song Love S	I like	2	0	0	1
14	74a76f6e0a	My Sharpie is running DANGERou	DANGERously	1	1	0	0
15	04dd1d2e34	i want to go to music tonight but i	lost	1	1	0	0
16	bbe3cbf620	test test from the LG enV2	test test from the LG enV2	0	0	1	0

Рис 4.2 Загальний вигляд нашого датасета із заповненими даними.

Рисунок 4.2 демонструє що, дані заповнені, та містять різноманітну інформацію яка не схоже з іншими. З різних набора даних з таблиці були узяті та проаналізовані у таблиці 4.1.

Таблиця 4.1 – Приклад даних датасета

№	Текст	Негативний	Нейтральний	Позитивний
1	Hell Yeah!	0	1	0
2	HAPPY MOTHERS DAY.!	0	0	1
3	my boss is bullying me...	1	0	0

Були отримані наступні результати аналізу, 1 приклад містить всього 2 слова із знаком оклику. Якщо проаналізувати їх, стане зрозуміло, що ніякого сенсу вони особливо не несуть. Вони не містять погані чи добрі слова і є простим ствердженням. Тому можна зробити висновок що це є нейтральним реченням.

На прикладі 2, спостерігаємо, що речення містить 3 слова, усі вони написанні із великих букв, мають знак оклику у кінці. Принаймні зараз можна зробити певний аналіз і зрозуміти, що можливо ці слова свідчать про певний стан, оскільки вже відчувається певна характеристика позитивних даних. При більш детальному огляді, стає зрозумілим, що ці слова направлені до когось і містять характеристику привітання із добрим почуттям. Тому можна зробити людський висновок, що це речення має позитивну характеристику.

Текст під номер 3, містить 5 різних логічно пов'язаних слів між собою. На цей раз в текстові на кінці присутні 3 крапки, що може свідчати про незакінчений текст або певну спогаду або сум враховуючи людський аналіз. Після аналізу усіх слів можна знайти 1 ключове яке містить головну інформацію про характеристику даного речення. Підсумовуючи, можна із 100% шансом сказати, що це негативне речення, оскільки містить характеристики насилля.

Був зроблений найпростіший аналіз із людської точки зору і виявлено вже дуже багато факторів, які можуть вплинути на кінцевий результат. Головне питання постає в наступному, а саме як передати цю інформацію нейронній мережі, навчити її й зробити це як умова ефективніші?

Перша проблема яку потрібно вирішити, це спростити підхід й зробити його зрозуміліший й ефективним з машинної точки зору. Для цього було зроблено токенизацію слів. Тобто необхідність взяти текст й перетворити їх на токени, використовуючи впроваджені бібліотеки.

Токенизація – це метод перетворення даних на певну послідовність цифр яка буде зрозумілішою для машини. Токени це одиниці вимірювання або певна послідовність цифр, які мають зміст.

Ось приклад слово ‘HAPPY’ може бути показано як цифра 85 чи слово ‘boss’ як цифра 344 і так далі. Але нажаль ці цифри не дають багато сенсу для моделі, тому їх потрібно перетворити на вектор. Тобто 344 буду перетворено як [0.1, 0.5, 0.0, ...] і ці певні значення набувають можливості передачі як певні атрибути до нейронної мережі і вже тоді вона зможе із ними працювати.

За допомогою LSTM шарів [43] надано можливість ефективно навчити нейронну мережу, бо відомо, що LSTM дуже добре вміє працювати із послідовними даними. На виході буде багато двоїчного коду тобто 0 чи 1 що будуть мапитися назад до наших станів.

Після цього доступна можливість серілізувати їх та отримати результати. Після цього необхідно зберегти дану модель і засотувати її у веб додатку за допомоги Gradio.

Майбутня архітектурна схема наведена у додатку В, де показано лише 2 головні частини, тобто Python у вигляді сервера та Gradio у вигляді клієнтської частини.

4.2 Початок написання програми

Перше, що буде необхідно зробити – це встановити необхідні бібліотеки для роботи, які будуть працювати із даними та виконувати інші завдання. Приклад встановлення 1 із важливих бібліотек наведено на рисунку 4.3.

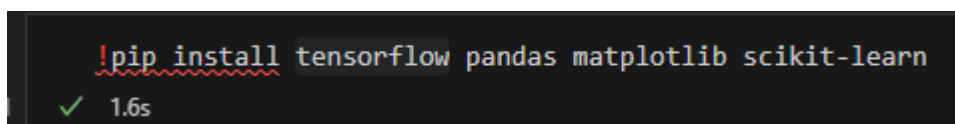


Рис 4.3 Встановлення бібліотек.

Після виконання цієї команди будуть встановлені необхідні пакети для роботи із такими бібліотеками як tensorflow, які у свою чергу працюють із моделями нейронної мережі.

Після того як бібліотеки було успішно завантажено, наступним кроком буде їх підключити/імпортувати до проекту для подальшої роботи як це показано на рисунку 4.4.

```
import os
import pandas as pd
import tensorflow as tf
import numpy as np
```

[80] ✓ 0.0s

Рис 4.4 Підключення бібліотек.

Після того як було підключено бібліотеку os та pandas, тепер система має змогу працювати із Excel\CSV форматами. Тому наступними кроками буде зчитування доробленого та виправленого датасету і спроба вичитати перші 5 значень, приклад якого можна спостерігати на рис 4.5 та рис 4.6.

```
# load data from csv file
df = pd.read_csv(os.path.join('values', 'Tweets4_2.csv'))
```

[81] ✓ 0.0s

Рис 4.5 Зчитування датасету з файлу

```
df.head()
```

[82] ✓ 0.0s

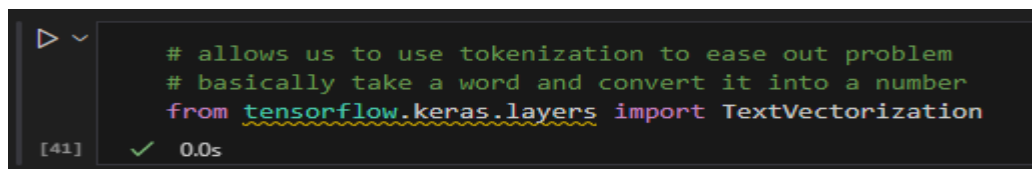
	textID	text	selected_text	sentiment	negative	neutral	positive
0	cb774db0d1	I'd have responded, if I were going	I'd have responded, if I were going	0	0	1	0
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	1	1	0	0
2	088c60f138	my boss is bullying me...	bullying me	1	1	0	0
3	9642c003ef	what interview! leave me alone	leave me alone	1	1	0	0
4	358bd9e861	Sons of ****, why couldn't they put them on t...	Sons of ****,	1	1	0	0

Рис 4.6 Зчитування перших 5 даних з завантаженого датасету

Як показують результати, вдалося запарсити файл та прочитати з нього дані. Наступним нашим кроком буде етап попередньої обробки.

4.3 Попередня обробка даних для моделі

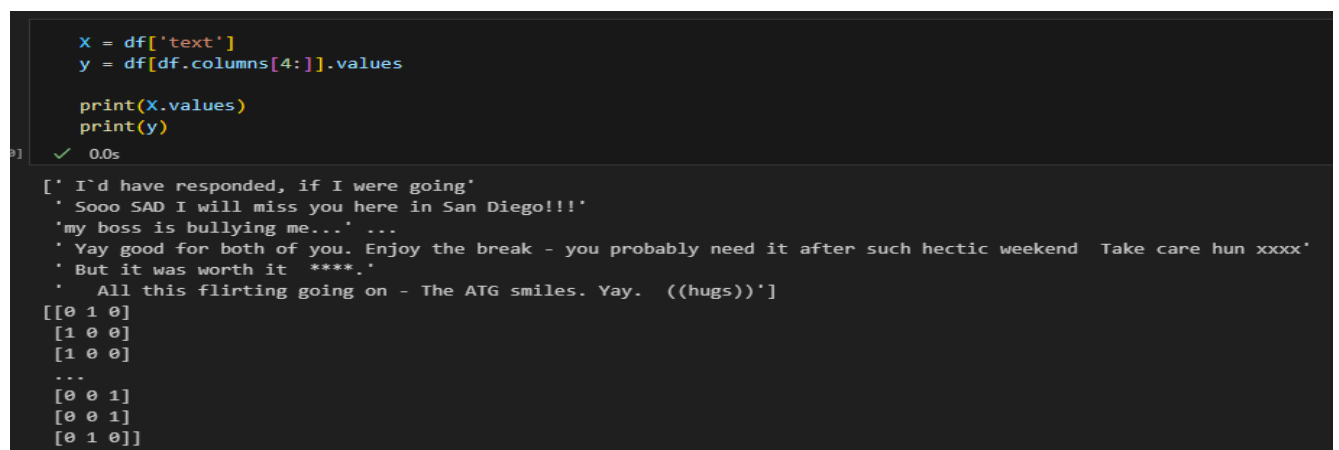
Настала необхідність зробити певні дії, перед тим як почати навчати модель. Тобто в першу чергу потрібно підготувати дані для ефективного навчання моделі. Для початку необхідно імпортувати певні додаткові бібліотеки, як це показано на рисунку 4.7.



```
[41]: # allows us to use tokenization to ease out problem
# basically take a word and convert it into a number
from tensorflow.keras.layers import TextVectorization
```

Рис 4.7 Підключення додаткової бібліотеки

Після підключення бібліотеки настає необхідність створити вектор який буде працювати із підготовленими словами. Для початку необхідно витягнути лише необхідні значення та впевнитися в їх правильності і після того почати створювати векторайзер [44] для адаптації даного тексту, як це показано на рисунку 4.8 та створення векторайзера на рисунку 4.9.



```
x = df['text']
y = df[df.columns[4:]].values

print(x.values)
print(y)
```

```
[ ' I'd have responded, if I were going'
' Sooo SAD I will miss you here in San Diego!!!'
' my boss is bullying me...' ...
' Yay good for both of you. Enjoy the break - you probably need it after such hectic weekend Take care hun xxxx'
' But it was worth it ****.'
' All this flirting going on - The ATG smiles. Yay. ((hugs))']
```

```
[[0 1 0]
 [1 0 0]
 [1 0 0]
 ...
 [0 0 1]
 [0 0 1]
 [0 1 0]]
```

Рис 4.8 Перевірка певних значень

```

# number of words in the vocab
# the more words the effective our model will be
MAX_FEATURES = 200000
16] ✓ 0.0s

vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                               # maximum length of our sentence will be
                               output_sequence_length=1800,
                               output_mode='int')
17] ✓ 0.0s

# we need to adapt our vectorizer to our words
vectorizer.adapt(X.values.tolist())
18] ✓ 0.8s

```

Рис 4.9 Створення векторайзера

Можна спостерігати, що спершу було створено змінну, яка буде позначати максимальну кількість слів в даному словнику. Чим більше значення, тим більше буде дана модель і тим більше буде якість.

Далі, необхідно створити вектор який буде працювати із реченнями. Для цього є потреба викликати клас `TextVectorization` і передати в нього максимальну кількість словника, далі передати максимальну кількість слів даного речення, і знову чим більше тим краще, але час на навчання також буде збільшуватись. І останнє, необхідно вказати що вихідні дані будуть типу `int`, тобто цифри.

Після усіх запроваджених дій буде передано підготовлений текст і започатковано процес навчання.

Після усіх цих дій необхідно викликати додатковий метод щоб перевірити як були перетворені певні речення і перевірити які саме значення повертаються. Подібні дії продемонстровані на рисунках 4.10 та 4.11.

```
vectorizer.get_vocabulary()
[89] ✓ 0.0s
... [' ',
     '[UNK]',
     'i',
     'to',
     'the',
     'a',
     'my',
     'and',
     'you',
     'it',
     'is',
     'in',
     'for',
     'of',
     'im',
     'on',
     'me',
     'have',
     'so',
     'that',
     'but',
     'just',
     'with',
     'day',
     'be',
     ...
     'case',
     'alot',
     'xoxo',
     'wearing',
     ...]
```

Рис 4.10 Перевірка розбиття слів всередині вектора на слайси

```
vectorizer('Hi there old friend')[:6]
[90] ✓ 0.0s
... <tf.Tensor: shape=(6,), dtype=int64, numpy=array([299, 64, 216, 253, 0, 0], dtype=int64)>
```

Рис 4.11 Перевірка розбиття слів всередині вектора із значеннями

Після того як векторайзер було навчено, наступним кроком буде праця із надходим текстом, тому є потреба додатково передати текст щоб він його векторизував, як це показано на рисунку 4.12.

```
vectorized_text = vectorizer(X.values.tolist())
print(vectorized_text)
```

[91] ✓ 0.2s

... tf.Tensor(
 [[293 17 15183 ... 0 0 0]
 [413 115 2 ... 0 0 0]
 [6 1335 10 ... 0 0 0]
 ...
 [225 31 12 ... 0 0 0]
 [20 9 28 ... 0 0 0]
 [29 30 6478 ... 0 0 0]], shape=(27481, 1800), dtype=int64)

Рис 4.12 Перевірка розбиття слів всередині нашого вектора

Векторайзер успішно опрацював та перевів кожне слово в цифру з наданого тексту.

Наступним кроком буде створення конвеєра яке повинно об'легшити життя моделі. Головна задача даного конвеєра це опрацювання даних які не можуть поміститися пам'ять.

На першому етапі створюється сет для даного конвеєра за допомогою функцій з бібліотеки датасету, тобто просто передаємо присутні дані для опрацювання.

Далі необхідно запровадити кешування даних, після цього робиться їх перемішання, встановлюючи яким великим буде буфер по величині потрібен бути. Далі вони розбиваються на невеличкі сети по 16 слів.

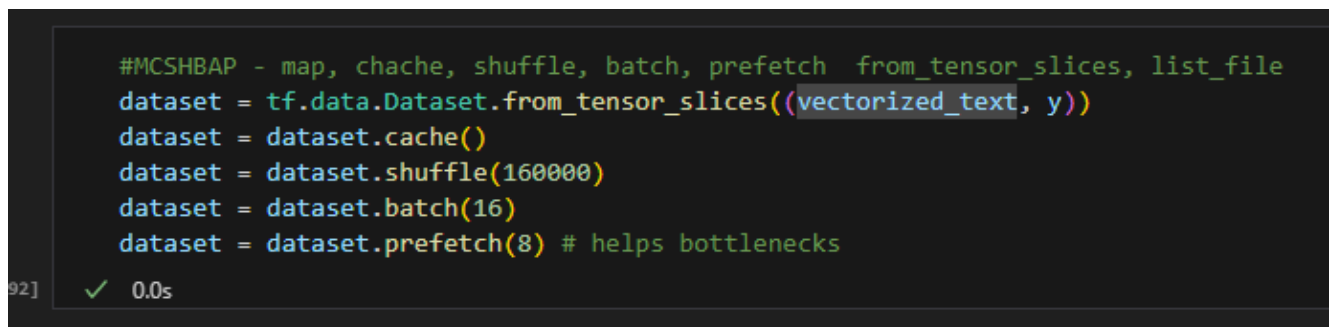
Після, використовується метод `prefetch(8)` на об'єкті `dataset`. Метод `prefetch` використовується для попереднього завантаження елементів датасету в пам'ять комп'ютера перед тим, як вони будуть фактично потрібні для навчання моделі чи використання в іншому контексті.

Аргумент 8 вказує, що система повинна попередньо завантажити 8 елементів датасету.

Це може бути корисно для запобігання "заторів" чи затримок, які можуть виникнути під час навчання моделі, коли дані потрібні для наступного батча, а вони ще не завантажені в пам'ять.

Використання `prefetch` може допомогти у використанні часу, більш ефективно та підвищити продуктивність моделі, оскільки вона не буде чекати на завантаження даних, а може продовжувати обробку та навчання.

Усі описані дії продемонстровані на рисунку 4.13.



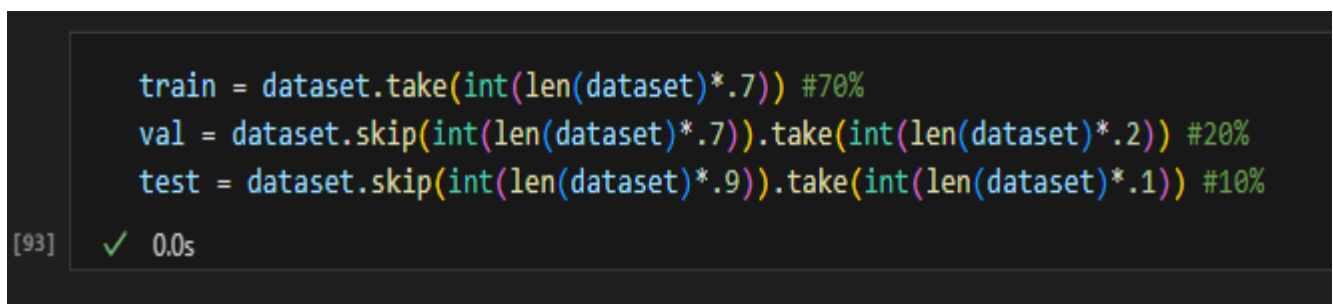
```
#MCSHBAP - map, cache, shuffle, batch, prefetch from_tensor_slices, list_file
dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
dataset = dataset.cache()
dataset = dataset.shuffle(160000)
dataset = dataset.batch(16)
dataset = dataset.prefetch(8) # helps bottlenecks
```

92] ✓ 0.0s

Рис 4.13 Створення конвеєра і робота із ним

І в самому кінці датасет поділяється або дані з конвеєра на дані для навчання тобто 70%, дані для валідації даних – 20% і останнє це тестові дані які становлять 10%.

Загальна пропорція буде складати 70\20\10. Це показано на рисунку 4.14.



```
train = dataset.take(int(len(dataset)*.7)) #70%
val = dataset.skip(int(len(dataset)*.7)).take(int(len(dataset)*.2)) #20%
test = dataset.skip(int(len(dataset)*.9)).take(int(len(dataset)*.1)) #10%
```

[93] ✓ 0.0s

Рис 4.14 Розбиття датасета для тестування, валідації та тестування

4.4 Створення послідовної моделі

Наступний етап полягає в створенні послідовної моделі, оскільки вона найкраще підходить в даному випадку для праці із послідовними даними.

Спочатку підключаються потрібні бібліотеки, після цього викликається клас який працює з послідовою моделлю, тобто надає можливість накладати різні рівні.

Після цього активується вбудований рівень, який зазвичай використовується для обробки та вивчення вставок слів у завданнях обробки природної мови. У цьому випадку він ініціалізується `MAX_FEATURES+1` як вхідний розмір (розмір словника) і 32 як вихідний розмір (розмір вбудовування). Він перетворює цілі індекси, що представляють слова, у щільні вектори фіксованого розміру - 32 у цьому випадку.

Далі прописується +1 оскільки для невідомих слів буде окремо з ними працювати.

Після цього додається двонаправлений шар LSTM. Двонаправлений LSTM обробляє вхідну послідовність як у прямому, так і в зворотному напрямках, захоплюючи контекстну інформацію з минулого та майбутнього.

Шар має 32 одиниці, а функцією активації використовується гіперболічний тангенс (`tanh`).

Наступні шари діють як екстрактори ознак і вносять нелінійність у модель.

Останній шар із 3 одиницями, що представляють кількість класів у завданні класифікації. Використана функція активації сигмоїдна, яка підходить для задач двійкової класифікації, тобто в нашому випадку.

Якби це було багатокласове завдання класифікації, функція активації `softmax` могла б бути більш доречною.

Після того як все це закінчиться, буде скомпільована дана модель використовуючи функцію `BinaryCrossentropy`, тому що йде праця з 3 наборами бінарних даних, які будуть виходити.

Тому це є найоптимальнішим варіантом серед усього в даному випадку.

Після того як було успішно створено послідовну модель і зроблено її, то мається можливість переглянути її внутрішню схему моделі, яку надано на рисунку 4.15.

Наступним кроком буде її навчання. Навчання починається з 25 епохами, як це показано на рисунку 4.16

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Bidirectional, Dense, Embedding

✓ 0.0s

model = Sequential()
# Create the embedding layer
model.add(Embedding(MAX_FEATURES+1, 32))
# Bidirectional LSTM Layer
model.add(Bidirectional(LSTM(32, activation='tanh')))
# Feature extractor Fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
# Final layer
model.add(Dense(3, activation='sigmoid'))

✓ 0.3s

model.compile(loss='BinaryCrossentropy', optimizer='Adam')

✓ 0.0s
```

Рис 4.15 Створення послідовної моделі та накладання різних шарів

Також, ми надається можливість візуально уявити дану архітектуру нейронної мережі, для більш зручного розуміння, вона наведена в додатку Б а її функціональна схема, яка показує принцип роботи наведена в додатку Г.

Завдяки даним схемам, здобувається уявлення про роботу, яка відбувається з середини та як сама працює двонаправлена LSTM мережа.

```

history_2 = model.fit(train, epochs=25, validation_data=val)
✓ 350m 16.4s

Epoch 1/25
1202/1202 [=====] - 711s 590ms/step - loss: 0.0634 - val_loss: 0.0468
Epoch 2/25
1202/1202 [=====] - 954s 793ms/step - loss: 0.0579 - val_loss: 0.0423
Epoch 3/25
1202/1202 [=====] - 1019s 848ms/step - loss: 0.0496 - val_loss: 0.0452
Epoch 4/25
1202/1202 [=====] - 1041s 866ms/step - loss: 0.0415 - val_loss: 0.0369
Epoch 5/25
1202/1202 [=====] - 1038s 864ms/step - loss: 0.0408 - val_loss: 0.0287
Epoch 6/25
1202/1202 [=====] - 1037s 862ms/step - loss: 0.0351 - val_loss: 0.0305
Epoch 7/25
1202/1202 [=====] - 1034s 861ms/step - loss: 0.0332 - val_loss: 0.0246
Epoch 8/25
1202/1202 [=====] - 978s 813ms/step - loss: 0.0292 - val_loss: 0.0217
Epoch 9/25
1202/1202 [=====] - 644s 536ms/step - loss: 0.0261 - val_loss: 0.0244
Epoch 10/25
1202/1202 [=====] - 902s 751ms/step - loss: 0.0235 - val_loss: 0.0156
Epoch 11/25
1202/1202 [=====] - 950s 791ms/step - loss: 0.0232 - val_loss: 0.0205
Epoch 12/25
1202/1202 [=====] - 892s 742ms/step - loss: 0.0209 - val_loss: 0.0287
Epoch 13/25
1202/1202 [=====] - 750s 624ms/step - loss: 0.0200 - val_loss: 0.0114
...
Epoch 24/25
1202/1202 [=====] - 623s 519ms/step - loss: 0.0089 - val_loss: 0.0061
Epoch 25/25
1202/1202 [=====] - 631s 525ms/step - loss: 0.0104 - val_loss: 0.0040
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```

Рис 4.16 Показ навчання моделі використовуючи 25 епох

Можна спостерігати як модель успішна виконала тренування в розмірі 25 епох/разів, а її результати навчання наведені на рисунку 4.17

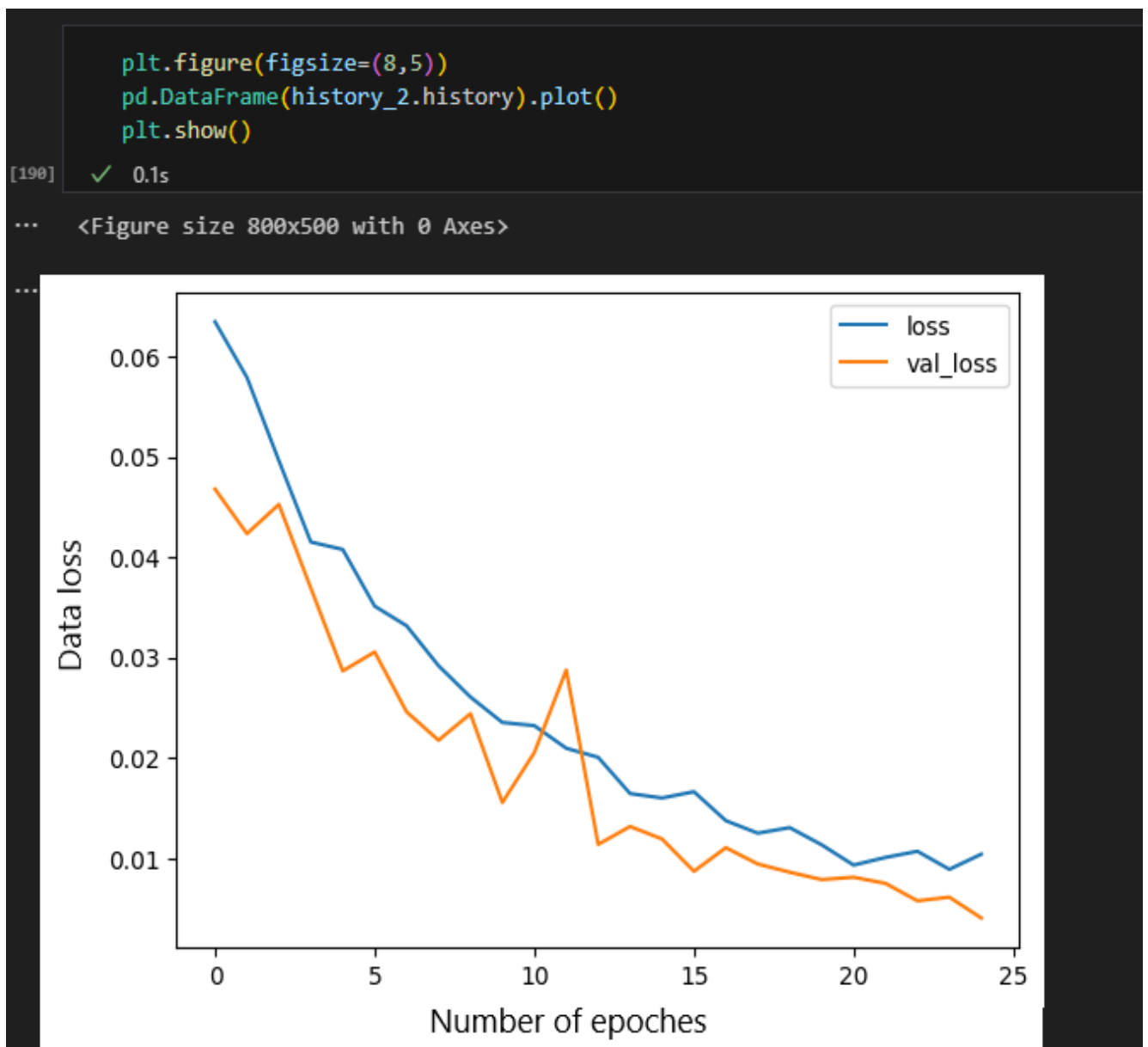


Рис 4.17 Діаграма втрати точності під час навчання

Надається можливість аналізу де від кожного наступного навчання такі показники як `loss` та `val_loss` зменшувалися, що свідчить про те, що це є позитивним знаком, оскільки модель становилися більш точною з кожною рівнем.

Спостерігається невеличкий скачок який стався, але після цього стабілізувався і не мав ніяких проблем в подальшому.

Підсумовуючи, результат навчання свідчить про те, що модель прогресує в навчанні зі зменшенням втрат даних протягом епох.

4.5 Прогнозування даних використовуючи модель

Тепер маючи модель, є можливість спробувати прогнозувати. Для ефективності буде узятє речення, яке не зустрічалось всередині створеного датасету для його векторизації як показана на рисунку 4.18.

```
input_text = vectorizer('I like when blue cars next to me')
input_text
```

[220] ✓ 0.0s

... <tf.Tensor: shape=(1800,), dtype=int64, numpy=array([2, 37, 81, ..., 0, 0, 0], dtype=int64)>

Рис 4.18 Приклад векторизації текста

Передивляючись результат після векторизації, створився лист з цифрами, які позначають слова. Після цього надається можливість викликати дану модель і передати туди нові дані для їх обробки як на рисунку 4.19.

```
res = model.predict(np.expand_dims(input_text, 0))
res
```

[234] ✓ 0.1s

... 1/1 [=====] - 0s 81ms/step

... array([[0.00174553, 0.02736186, 0.9617492]], dtype=float32)

Рис 4.19 Прогнозування нових даних

Результати свідчать, що нові дані були правильно оброблені і правильно видані результати характеристики даного текста, оскільки воно є позитивним.

Для зручності можна уявити всю схему за допомогою додатка Д, де описується блок-схема алгоритму, який було застосовано.

Також, в якості прикладу є можливість обрати батчи зі створеного тестового датасету і перевірити їх на стани, як це показано на рисунку 4.20.

```

batch_X, batch_y = test.as_numpy_iterator().next()
[231] ✓ 0.1s

(model.predict(batch_X) > 0.5).astype(int)
[232] ✓ 0.1s

... 1/1 [=====] - 0s 120ms/step

... array([[0, 0, 1],
          [0, 0, 1],
          [0, 1, 0],
          [0, 0, 1],
          [1, 0, 0],
          [1, 0, 0],
          [1, 0, 0],
          [0, 1, 0],
          [1, 0, 0],
          [0, 0, 1],
          [1, 0, 0],
          [0, 1, 0],
          [1, 0, 0],
          [1, 0, 0],
          [0, 1, 0],
          [0, 1, 0]])

```

Рис 4.20 Приклад перевірки станів 1 батча

4.6 Оцінка моделі

Наступний фрагмент коду обчислює показники точності, запам'ятовування та категоричної точності для моделі машинного навчання, застосованої до тестового набору даних.

Перебираючи пакети тестового набору даних, код генерує прогнози за допомогою моделі та коригує стани показників на основі справжніх і прогнозованих значень.

Цей процес конкретно визначає точність, запам'ятовуваність і точність продуктивності моделі на тестових даних, як це можна побачити на рисунку 4.21.

Значення прецизійності, запам'ятовування та точності постійно оновлюються для кожної партії, що зрештою сприяє комплексній оцінці ефективності моделі щодо цих показників.

```

[197] ✓ 0.0s
from tensorflow.keras.metrics import Precision, Recall, CategoricalAccuracy

pre = Precision()
re = Recall()
acc = CategoricalAccuracy()

[198] ✓ 0.0s

for batch in test.as_numpy_iterator():
    # Unpack the batch
    X_true, y_true = batch
    # Make a prediction
    yhat = model.predict(X_true)

    # Flatten the predictions
    y_true = y_true.flatten()
    yhat = yhat.flatten()

    pre.update_state(y_true, yhat)
    re.update_state(y_true, yhat)
    acc.update_state(y_true, yhat)

[199] ✓ 28.1s
... 1/1 [=====] - 0s 103ms/step
1/1 [=====] - 0s 106ms/step
1/1 [=====] - 0s 110ms/step
1/1 [=====] - 0s 107ms/step
1/1 [=====] - 0s 100ms/step
1/1 [=====] - 0s 103ms/step
1/1 [=====] - 0s 101ms/step
1/1 [=====] - 0s 102ms/step
1/1 [=====] - 0s 100ms/step
1/1 [=====] - 0s 103ms/step

```

Рис 4.21 Оцінка моделі через батчі

Було обчислено показники точності, запам'ятовування та точності для моделі машинного навчання на тестовому наборі даних.

Модель отримує поточні значення точності, запам'ятовування та точності за допомогою методу `result()` для кожного показника, а потім друкує ці значення у форматованому рядку. Роздрукований результат показує продуктивність моделі з точки зору точності, запам'ятовування та точності тестових даних. Приклад наведений на рисунку 4.22.

```

[239] ✓ 0.0s
print(f'Precision: {pre.result().numpy()}, Recall:{re.result().numpy()}, Accuracy:{acc.result().numpy()}')

... Precision: 0.997806191444397, Recall:0.9974415302276611, Accuracy:0.5204678177833557

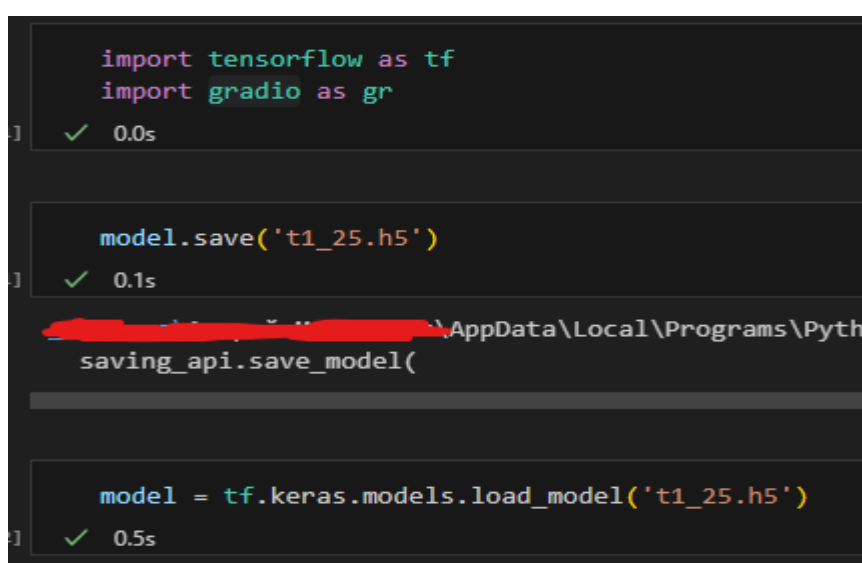
```

Рис 4.22 Загальна оцінка моделі

4.7 Тестування моделі

Після усіх успішних етапів прийшов час використати дану модель разом із веб додатком, який надасть можливість взаємодіяти з навченою моделлю, та буде надавати можливість протестувати її.

Першим кроком буде підключення додаткових бібліотек для збереження нашої моделі для подальшого використання, як це показано на рисунку 4.23.



```
import tensorflow as tf
import gradio as gr
✓ 0.0s

model.save('t1_25.h5')
✓ 0.1s

... \AppData\Local\Programs\Python

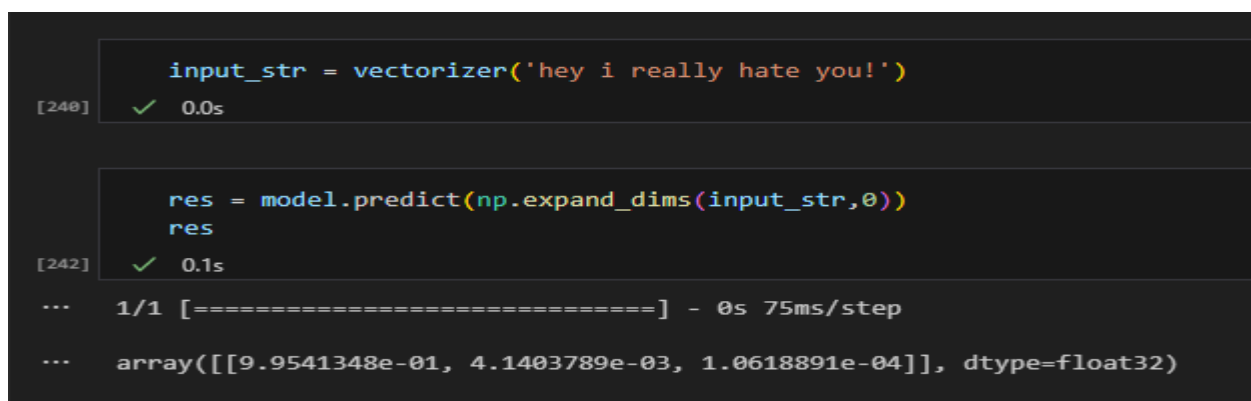
saving_api.save_model(

...

model = tf.keras.models.load_model('t1_25.h5')
✓ 0.5s
```

Рис 4.23 Зберігання та загрузка моделі

Наступним кроком буде повторне тестування загруженої моделі, як це показується на рисунку 4.24.



```
[240] input_str = vectorizer('hey i really hate you!')
✓ 0.0s

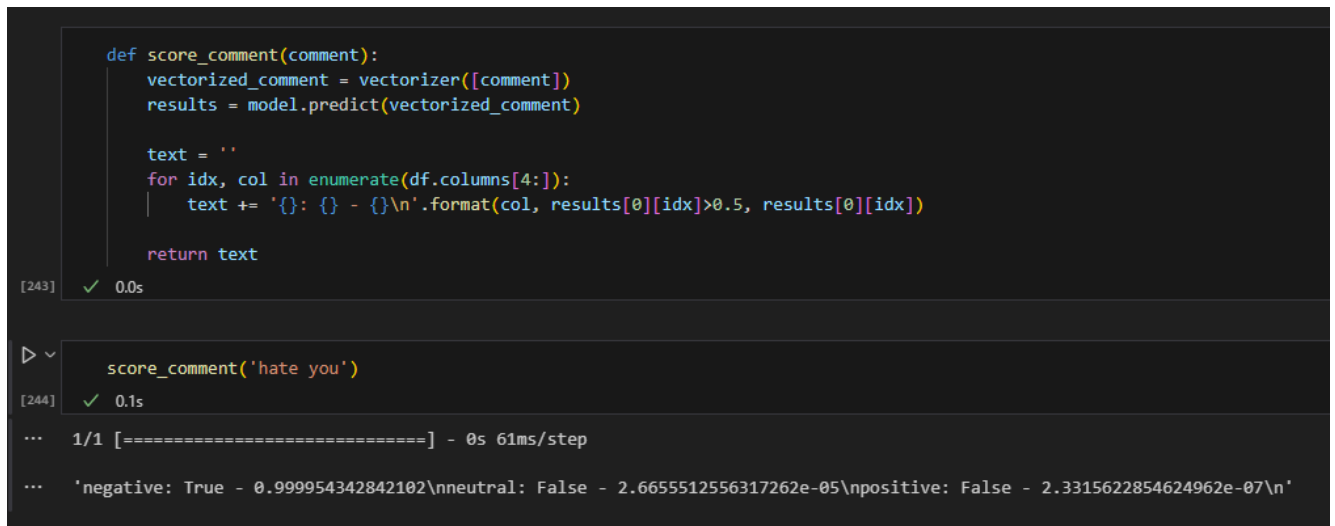
[242] res = model.predict(np.expand_dims(input_str,0))
res
✓ 0.1s

... 1/1 [=====] - 0s 75ms/step

... array([[9.9541348e-01, 4.1403789e-03, 1.0618891e-04]], dtype=float32)
```

Рис 4.24 Тестування загруженої моделі

Можна побачити, що модель працює вірно. Наступним кроком, буде написання додаткового методу для оцінки даних які будуть надходити на веб застосунок, як це показано на рисунку 4.25.



```
def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} - {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
```

[243] ✓ 0.0s

```
score_comment('hate you')
```

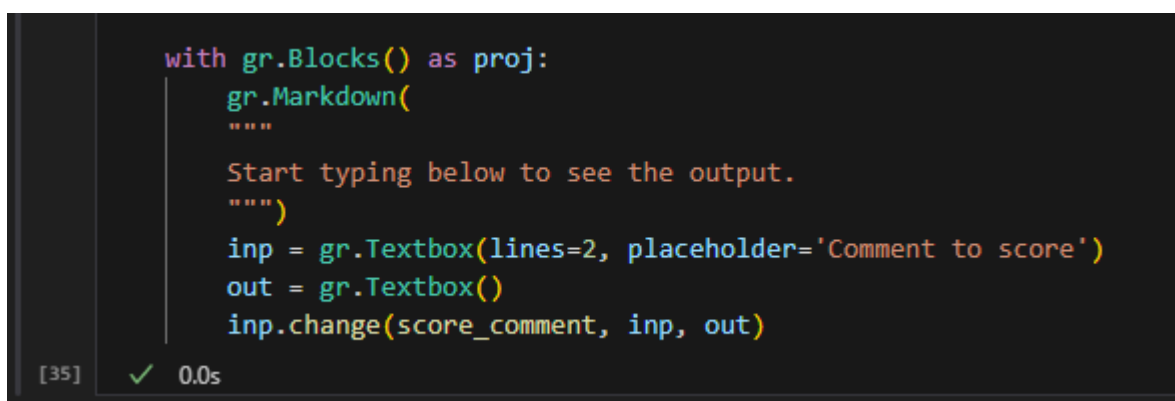
[244] ✓ 0.1s

... 1/1 [=====] - 0s 61ms/step

... 'negative: True - 0.999954342842102\nneutral: False - 2.6655512556317262e-05\npositive: False - 2.3315622854624962e-07\n'

Рис 4.25 Тестування методу для веб застосунку

Після успішного тестування буде запущено Gradio із використанням цього методу із середини як на рисунку 4.26, та існує необхідність протестувати даний додаток із використанням моделі для характеристики даних, як на рисунку 4.27 та 4.28.



```
with gr.Blocks() as proj:
    gr.Markdown(
        """
        Start typing below to see the output.
        """
    )
    inp = gr.Textbox(lines=2, placeholder='Comment to score')
    out = gr.Textbox()
    inp.change(score_comment, inp, out)
```

[35] ✓ 0.0s

Рис 4.26 Запуск веб застосунку Gradio

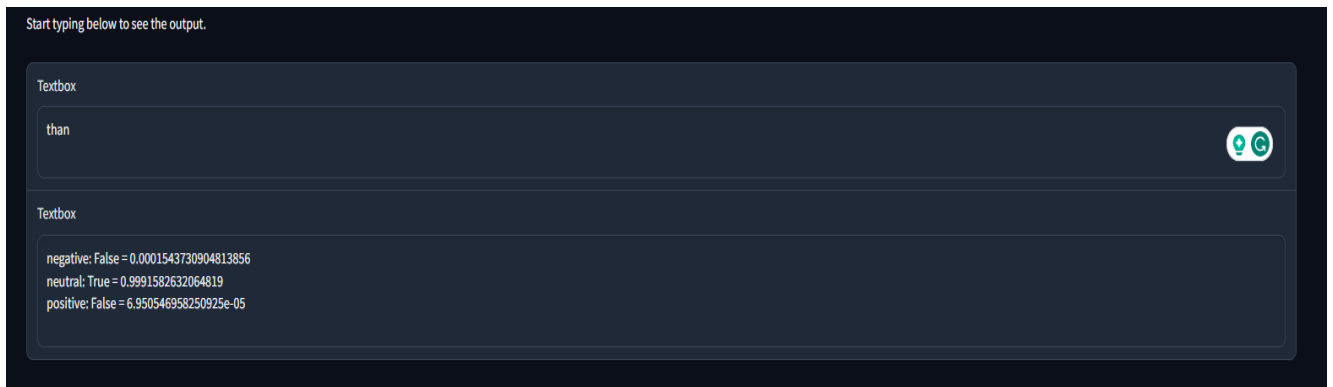


Рис 4.27 Приклад використання Gradio разом із моделлю з першими випадковими значеннями від юзера

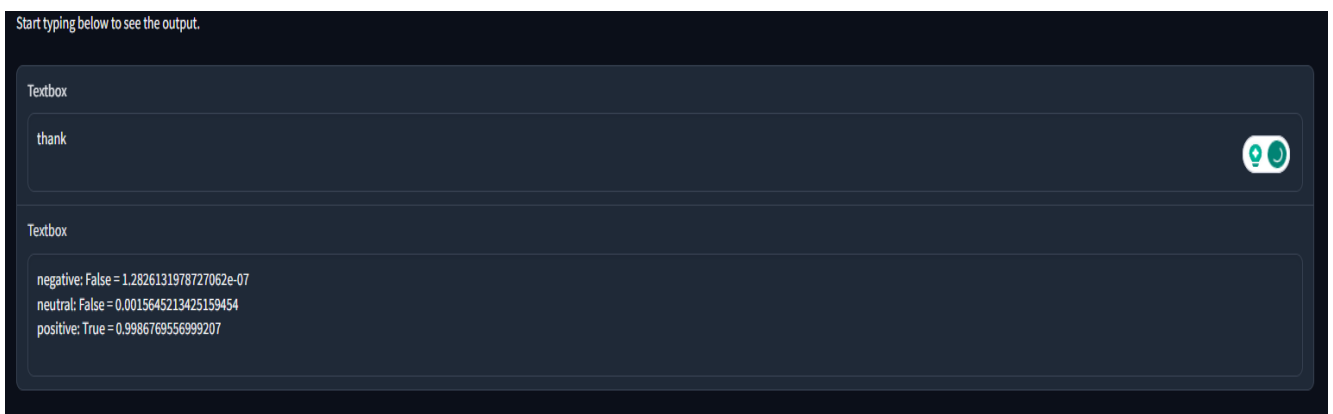


Рис 4.28 Приклад використання Gradio разом із моделлю з другими випадковими значеннями від юзера

Спостерігаємо успішність написання та тестування послідовної моделі із різними шарами, яка працює в 2 напрямках для векторного аналізу даних. Було успішно навчено модель, та запущено веб додаток разом із його тестуванням.

Також під час навчання моделі було додатково збережено її модель до внутрішнього складу проєкта для подальшого користування.

Мається можливість додатково виділити 2 головних класи, які постійно застосовувалися під час навчання а також під час практичного тесту, це класи Dataset та TextVectorizer, які наведені в додатку А.

4.8 Аналіз результатів

Перед початком тестування було визначено, що буде оцінюватися результати 3 різних моделей, кожна з яких має різну кількість епох, що покаже ефективність навчання. Будуть присутні моделі із 4, 10 та 25 епохами [45].

Для перевірки ефективності моделей будуть використанні 4 різні фрази:

- hey I really hate you! / хей, я тебе справді ненавиджу!;
- I am not sure about this / Я не впевнений у цьому;
- I knew you were right / Я знав, що ти правий;
- that poor kid is stupid / та бідна дитина дурна.

Спочатку буде проведено аналіз із використанням людського фактору, тобто із людської точки зору.

Перша фраза, вона є направлена на певну особу, містить чітке направлення, головна характеристика якого є негативна частина. Цю частину не можна сприймати як нейтральний або позитивний напрямок, оскільки чітко прослідковуються негативний напрямок. Тому необхідно зробити висновок, що дана фраза є негативною.

Друга фраза, містить слова, які не викликають певні направлення, ця фраза є розмитою та вирваною із контекста, тому необхідно робити її огляд із точки зору саме цієї фрази. З погляду людини, ця фраза є суто нейтральною, її не можна сприймати як негативно чи позитивною.

Третя фраза, є направленою до певної персони, необхідно зауважити що дана фраза не містить негативних чи позитивних напрямків, також відчувається, що вона вирвана з контекста. Робиться висновок, що ця фраза є більш нейтральною аніж позитивною чи негативною.

Остання четверта фраза є негативним висновком, направлена в розмові. Можливо побачити певні слова, такі як 'бідна' та 'тупа', які формують дуже негативну думку щодо цього речення. Тому є можливість точно сказати, що дана фраза є негативною, аніж нейтральною чи позитивною.

Все це можливо побачити та зрозуміти більш детально за допомоги додатка И, а також для більш зрозумілого потоку дій, які юзер може запровадити з моделлю визначається в додатку Ж.

Тепер буде перевірено результати роботи нейронної моделі із 4, 10 та 25 епохами навчання та порівняно результати аналізу текста.

Спочатку буде перевірено результати роботи моделі із 4 епохами, оцінка результатів буде показано на рисунках з 4.29 по 4.32

```
[17] ✓ 0.0s
input_str = vectorizer('hey i really hate you!')

[18] ✓ 0.6s
res = model.predict(np.expand_dims(input_str,0))
res
... 1/1 [=====] - 1s 612ms/step
... array([[0.9760974 , 0.02044651, 0.0027498 ]], dtype=float32)

[19] ✓ 0.0s
def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text

[20] ✓ 0.1s
score_comment('hey i really hate you!')
... 1/1 [=====] - 0s 61ms/step
... 'negative: True = 0.9760974049568176\nneutral: False = 0.02044651098549366\npositive: False = 0.0027497983537614346\n'
```

Рис 4.29 Перевірка першої фрази за допомоги моделі із 4 епохами

```

input_str = vectorizer('I am not sure about this')
[21] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[22] ✓ 0.0s
... 1/1 [=====] - 0s 56ms/step
... array([[0.20106946, 0.74910486, 0.05010027]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[23] ✓ 0.0s

score_comment('I am not sure about this')
[25] ✓ 0.1s
... 1/1 [=====] - 0s 58ms/step
... 'negative: False = 0.20106945931911469\nneutral: True = 0.7491048574447632\npositive: False = 0.05010027438402176\n'

```

Рис 4.30 Перевірка другої фрази за допомоги моделі із 4 епохами

```

input_str = vectorizer('I knew you were right')
[26] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[27] ✓ 0.1s
... 1/1 [=====] - 0s 59ms/step
... array([[0.01826888, 0.43886632, 0.5207402 ]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[28] ✓ 0.0s

score_comment('I knew you were right')
[29] ✓ 0.1s
... 1/1 [=====] - 0s 63ms/step
... 'negative: False = 0.018268879503011703\nneutral: False = 0.4388663172721863\npositive: True = 0.5207402110099792\n'

```

Рис 4.31 Перевірка третьої фрази за допомоги моделі із 4 епохами

```

input_str = vectorizer('that poor kid is stupid')
[36] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[37] ✓ 0.1s
... 1/1 [=====] - 0s 57ms/step
... array([[9.9908191e-01, 9.1630709e-04, 2.6903745e-05]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[38] ✓ 0.0s

score_comment('that poor kid is stupid')
[40] ✓ 0.1s
... 1/1 [=====] - 0s 58ms/step
... 'negative: True = 0.9990819096565247\nneutral: False = 0.0009163070935755968\npositive: False = 2.6903744583250955e-05\n'

```

Рис 4.32 Перевірка четвертої фрази за допомоги моделі із 4 епохами

Можна спостерігати що, модель із 4 епохами надали дані, які є неточними, оскільки третю фразу вона розпізнала більше як позитивну, аніж нейтральну. Перше речення було негативне як ~97%, друге речення як ~75% нейтральне, що більш менш є правильним, але все ж таки ~25% як інше, що позбужує про думку, що це не є акуратним результатом. Третя фраза була розпінена більше як позитивна із ~52%, та ~43% як нейтральне, що вказує про неточність аналізу. Остання, четверта фраза має показник у ~99.9%, що є вкрай точним показником нашого попереднього людського аналізу. Тому, робимо висновок, що дана модель може оцінювати фрази з доволі великою точністю, але все ж таки допускає помилки.

Тепер було перевірено ті самі фрази але вже використовуючи модель із 10 епохами. Дані перевірки будуть надані на рисунках з 4.33 по 4.36 включно.

```

input_str = vectorizer('hey i really hate you')
[42] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[43] ✓ 0.5s

... 1/1 [=====] - 1s 502ms/step

... array([[0.97144943, 0.01595552, 0.00660272]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[44] ✓ 0.0s

▷ ▾ score_comment('hey i really hate you')
[45] ✓ 0.1s

... 1/1 [=====] - 0s 80ms/step

... 'negative: True = 0.9714494347572327\nneutral: False = 0.015955520793795586\npositive: False = 0.006602723151445389\n'

```

Рис 4.33 Перевірка першої фрази за допомоги моделі із 10 епохами

```

input_str = vectorizer('I am not sure about this')
[46] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[47] ✓ 0.1s

... 1/1 [=====] - 0s 60ms/step

... array([[0.0338836 , 0.9298917 , 0.02840133]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[48] ✓ 0.0s

▷ ▾ score_comment('I am not sure about this')
[49] ✓ 0.1s

... 1/1 [=====] - 0s 59ms/step

... 'negative: False = 0.03388359770178795\nneutral: True = 0.9298917055130005\npositive: False = 0.028401333838701248\n'

```

Рис 4.34 Перевірка другої фрази за допомоги моделі із 10 епохами

```

input_str = vectorizer('I knew you were right')
[50] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[51] ✓ 0.1s
... 1/1 [=====] - 0s 62ms/step
... array([[0.01123468, 0.70698667, 0.2566846 ]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[52] ✓ 0.0s

score_comment('I knew you were right')
[53] ✓ 0.1s
... 1/1 [=====] - 0s 56ms/step
... 'negative: False = 0.01123468391597271\nneutral: True = 0.706986665725708\npositive: False = 0.2566846013069153\n'

```

Рис 4.35 Перевірка третьої фрази за допомоги моделі із 10 епохами

```

input_str = vectorizer('that poor kid is stupid')
[54] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[55] ✓ 0.1s
... 1/1 [=====] - 0s 61ms/step
... array([[9.9984169e-01, 6.3868531e-05, 1.1090618e-05]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[52] ✓ 0.0s

score_comment('that poor kid is stupid')
[56] ✓ 0.1s
... 1/1 [=====] - 0s 60ms/step
... 'negative: True = 0.9998416900634766\nneutral: False = 6.386853056028485e-05\npositive: False = 1.1090617590525653e-05\n'

```

Рис 4.36 Перевірка четвертої фрази за допомоги моделі із 10 епохами

На цей раз є можливість спостерігати що, модель із 10 епохами змогла розпізнати усі фрази правильно та із більшою точністю, що це все свідчить про загальне покращення аналізу тексту та його загальних показників таких як точність.

Перше речення було негативне як $\sim 97\%$, що свідчить про однакову оцінку цього речення із попереднім результатом. Поки що не можна сказати що саме ця фраза впливає на загальний результат, оскільки і так відомо що ця фраза є доволі негативною.

Друге речення як $\sim 93\%$ нейтральне, що є значним покращенням попереднього результат та свідчить про більш вже високу ефективність аналізу тексту. Спостерігається загальне підвищення аналізу нейтрального характеру тексту.

Третя фраза була розпізнана більше як нейтральна із показником у $\sim 70.7\%$, що вказує про правильність оцінки тексту та покращення його аналізу. Результат покращився на $\sim 27\%$, що є суттєвим покращенням аналізом тексту. Це свідчить про набагато краще розуміння моделі із 10 епохами для розпізнавання нейтральних відтінків тексту.

Остання, четверта фраза має показник у $\sim 99.99\%$, що є вкрай точним показником, але має лише несуттєві покращення показника, але суттєві для самої моделі, оскільки це вказує на загальне покращення результатів та аналізу тексту в цілому.

Є можливість зробити висновок, що дана модель може оцінювати фрази з доволі великою точністю, і вже не допускає помилки, або шанс їх допуску є малим.

Тепер необхідно перевірити ті самі фрази але вже використовуючи модель із 25 епохами. Результати перевірки будуть показані на рисунках 4.37-4.40.

Це повинно набагато покращати аналізу тексту та результати роботи даної моделі. Загальна діаграма послідовності буде сформована в додатку Е.

```

input_str = vectorizer('hey I really hate you!')
[66] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[67] ✓ 0.1s
... 1/1 [=====] - 0s 64ms/step
... array([[9.9541348e-01, 4.1403789e-03, 1.0618891e-04]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[64] ✓ 0.0s

> ✓ 0.1s
score_comment('hey I really hate you!')
[68] ✓ 0.1s
... 1/1 [=====] - 0s 67ms/step
... 'negative: True = 0.9954134821891785\nneutral: False = 0.004140378907322884\npositive: False = 0.00010618891246849671\n'

```

Рис 4.37 Перевірка першої фрази за допомоги моделі із 25 епохами

```

input_str = vectorizer('I am not sure about this')
[62] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[63] ✓ 0.1s
... 1/1 [=====] - 0s 69ms/step
... array([[5.2079428e-02, 9.5402580e-01, 5.8330852e-04]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[64] ✓ 0.0s

> ✓ 0.1s
score_comment('I am not sure about this')
[65] ✓ 0.1s
... 1/1 [=====] - 0s 56ms/step
... 'negative: False = 0.05207942798733711\nneutral: True = 0.9540258049964905\npositive: False = 0.0005833085160702467\n'

```

Рис 4.38 Перевірка другої фрази за допомоги моделі із 25 епохами

```

input_str = vectorizer('I knew you were right')
[69] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[70] ✓ 0.1s
... 1/1 [=====] - 0s 59ms/step
... array([[1.3450529e-04, 9.0651393e-01, 6.4647675e-02]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[64] ✓ 0.0s

> score_comment('I knew you were right')
[71] ✓ 0.1s
... 1/1 [=====] - 0s 55ms/step
... 'negative: False = 0.00013450528786052018\nneutral: True = 0.9065139293670654\npositive: False = 0.06464767456054688\n'

```

Рис 4.39 Перевірка третьої фрази за допомоги моделі із 25 епохами

```

input_str = vectorizer('that poor kid is stupid')
[58] ✓ 0.0s

res = model.predict(np.expand_dims(input_str,0))
res
[59] ✓ 0.5s
... 1/1 [=====] - 1s 509ms/step
... array([[1.0000000e+00, 3.9227310e-09, 1.3083757e-13]], dtype=float32)

def score_comment(comment):
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[4:]):
        text += '{}: {} = {}\n'.format(col, results[0][idx]>0.5, results[0][idx])

    return text
[60] ✓ 0.0s

> score_comment('that poor kid is stupid')
[61] ✓ 0.1s
... 1/1 [=====] - 0s 57ms/step
... 'negative: True = 1.0\nneutral: False = 3.922731028893622e-09\npositive: False = 1.3083757439014826e-13\n'

```

Рис 4.40 Перевірка четвертої фрази за допомоги моделі із 25 епохами

На цей можливо спостерігати що, модель із 25 епохами змогла розпізнати усі фрази правильно із найвищою точністю. Перше речення було негативне як ~99.5%, що свідчить про покращення аналізу текста в порівнянні із попередніми результатами.

Друге речення визначено як ~95% нейтральне, що свідчить про невелику але все ж таки покращення результат у порівнянні із попередньої моделлю.

Третє речення тепер має показник у ~90.6%, що вказує на значення покращення у порівнянні із першою та другою моделлю.

Остання, четверта фраза має показник у 100%, це перший показник який має такий високий бал оцінювання речення.

Дивлячись на всі результати можна із великою вірогідністю сказати, що модель із 25 епохами, є найточнішою та найбільш надійною під час аналізу текста, що робить її фаворитом у використанні.

Далі, буду перевірено роботу автоматичного аналізу, тобто мінімальна кількість дій, яку юзеру необхідно зробити, для того щоб модель почала аналізувати даний текст. В даному випадку, юзерові буде достатньо лише ввести текст, для того щоб модель почала аналізувати його.

Аналіз та показ дій юзера показано на рисунках 4.41 – 4.43.

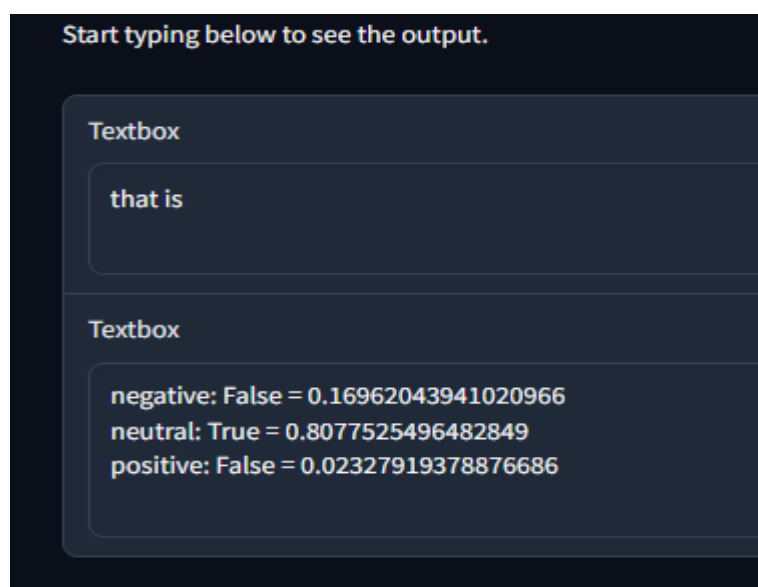


Рис 4.41 Неповний аналіз фрази

Start typing below to see the output.

Textbox

that is horr

Textbox

negative: True = 0.608680009841919
 neutral: False = 0.3795739710330963
 positive: False = 0.023619184270501137

Рис 4.42 Частковий аналіз фрази

Start typing below to see the output.

Textbox

that is horrible

Textbox

negative: True = 0.9999772906303406
 neutral: False = 2.2586527848034166e-05
 positive: False = 3.456844410720805e-08

Рис 4.43 Повний аналіз фрази

Підсумовуючи результати, можна сказати, що програма автоматично аналізує текст який юзер передає на вхід та обраховує його за трьома категоріями. В залежності від введених слів юзером можливо спостерігати, як змінюються певні коефіцієнти даного тексту, тим самим підтверджуючи ефективність роботи створеної моделі.

Висновки до розділу 4

Під час теоретичного планування та практичного написання було розроблено текстова нейрона мережа, яка виконує функції з оцінки текста, була приділена увага створенню унікального датасету, який містить в собі багато даних для аналізу.

Був створений проект на основі Python із застосування Jupiter Notebook. Була продемонстрована поетапна розробка нейронної мережі, із можливістю вичитки датасету, який було створено та модифіковано. Були продемонстровані методи пропроцесінгу даних для покращення ефективності роботи із ними, а саме векторизації та роботи із цифрами, які пізніше будуть закешовані, зміщені, поділені на батчи та використанні для створення послідовної моделі використовуючи функцію BinaryCrossentropy [46] та різні шари. Модель була навчена і перевірна на якість її роботи.

Була проведена оцінка роботи моделі, для цього було розпаковано створені батчи, де було перевірено кожне наступний батч за допомоги ітерації та використали різноманітні функції для цього.

У кінці, модель була протестована на різноманітних даних від юзера із застосування графічного додатка для зручності. Система була розроблена як автоматична, що дозволяє юзерові зосередитися лише на даних, що він віддає на вхід моделі.

Були протестовані різноманітні моделі із різною кількістю епох та порівняли результати, в результаті яких було виявлено, що найбільш ефективно виявилися модель із 25 епохами, що надавала найбільш точні результати аналізу текста.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї

Опис ідеї проекту є важливим фактором, тому необхідно проаналізувати ідею проекту, напрямки його застосування та потенційні вигоди для майбутніх користувачів. Опис наведений в таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекта

Зміст ідеї	Напрямки застосування	Вигода для користувача
Головна ідея полягає в створенні автоматизованої системи аналізу текста із легкою реалізацією для її подальшого покращення	Домашнє користування	За допомогою системи автоматичного аналізу текста, він може бути використано для соціальних аккаунтів як додатковий віджет для фільтрування текста чи як окрема програма для аналізу статті у вільному доступі
	Військова користування	Може бути використано військовими для модифікації існуючої моделі для аналізу великої кількості даних
	Бізнес і маркетинг	За допомоги даної моделі можна з легкістю аналізувати великі батчи даних для аналізу ефективності підходів маркетинга, в особливості коментарів від юзерів

У таблиці 5.2 показується порівняльний аналіз за технічними та економічними характеристиками в порівнянні із іншими існуючими системами на ринку, розкриваючи слабкі та сильні сторони.

Таблиця 5.2 – Визначення сильних, нейтральних та слабких характеристик ідеї проекту.

№	Технічні та економічні характеристики	Потенційні конкуренти та їх можливі продукти				W	N	S
		Власний проект (TC1)	TextInspector	Istio	QuestionPro			
1	Вузьке спеціалізоване користування функціоналом	+	+	+	+		+	
2	Можливість розширення базового функціоналу	+	-	-	-	+		
3	Модель надання сервісу	SaaS	IaaS	SaaS	SaaS		+	
4	Можливість безкоштовного користування	+	+	+	+		+	
5	Існуючий ліміт обмеження користування	+	-	-	+			+
6	Використання багатьох даних для аналізу тексту	-	+	+	+			+
7	Зручний користувацький інтерфейс	+	-	+	-	+		
8	Можливість інтеграції	-	-	+	+			+

Є можливість зробити наступний висновок, що є можливість продажу проекту як SaaS продукт, який буде надавати зручний та гнучкий інтерфейс в порівнянні із іншими, а також мати легку можливість розширення функціоналу. Також він буде надавати вузький функціонал, який буде використовувати вузьку специфіку методів і алгоритмів.

5.2 Технологічний аудит ідеї проекту

В таблиці 5.3 наведено технологічна здійсненність ідеї проекту.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№	Ідея	Технологічні засоби реалізації	Наявність	Доступність
1	Мова програмування	C#	Наявний	Доступний
		Python	Наявний	Доступний
		C++	Наявний	Доступний
Обрана мова програмування для реалізації ідеї проекту: Python				
2	Клієнтський інтерфейс для взаємодії	Angular	Наявний	Доступний
		React	Наявний	Доступний
		Gradio	Наявний	Доступний
Обраний клієнтський інтерфейс для користувача: Gradio				
3	Модель нейронної мережі	Послідовна	Наявний	Доступний
		Непослідовна	Наявний	Доступний
Обрана модель нейронної мережі ідеї проекту: Послідовна				
4	Тип послідовної мережі	LSTM	Наявний	Доступний
		Transforms	Наявний	Доступний
		RNN	Наявний	Доступний
Обраний тип послідовної мережі: LSTM				
5	Тип реалізації моделі	Дефолтна	Наявний	Доступний
		Модифікована	Власна реалізація	Власна реалізація
Обраний тип реалізації моделі: Модифікована				

Згідно результатам аналізу, можливо зробити висновок, що все стеки технологій є доступні, тому ідея проекту є можливою та технологічно доступною для запровадження.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Наступним головним кроком є визначення головних можливостей і небезпек, які можливо зустріти під час реалізації проекту. Даний аналіз повинен надати розуміння можливих ризиків.

Спочатку важливо зрозуміти попит, специфіку якого задокументовано в таблиці 5.4.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	\$8 млрд.
3	Динаміка ринку (якісна оцінка)	Легкий зріст
4	Наявність обмежень для входу (вказати характер обмежень)	Високі затрати на підтримку системи, схожі роботи, конкуренція, маркетинг
5	Специфічні вимоги до стандартизації та сертифікації	Сертифікати зі стандарту шифрування та безпеки
6	Середня норма рентабельності в галузі (або по ринку), %	Від 15% до 25%, в залежності від обраної галузі

Згідно результатом, на ринку є невелика кількість головних гравців, які мають налагоджену систему продаж та базу клієнтів. Враховуючи легкий зріст у динаміці ринку, можна зазначити, що навіть разом гравці генерують не маленьку суму прибутку при дохідності від 15 до 25 відсотків. Але не дивлячись на такі цифри, можна казати, що даний ринок є стабільним, оскільки маються клієнти чи компанії яким необхідні дані послуги для свого власного розвитку.

Не потрібно також забувати про певну сертифікацію, оскільки це також є важливою частини безпеки таких проектів.

Наступним пунктом необхідно розглянути зацікавлений або потенційно зацікавлений клієнтів, що можуть зацікавитися проектом. Ця інформація буде розглянута в таблиці 5.5

Таблиця 5.5 – Характеристика потенційних клієнтів

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Праця із великою кількістю текстових документів	Особи або приватні бізнеси, які потребують рішення для аналізу документів	Використовують сервіс для вилучення певних аналітичні дані	Можливість працювати із документами та правильно надавати вхідні параметри
2	Аналіз текстових даних	Особи, які зацікавленні в аналізі певних даних	Приватна або конференційна інформація	Можливість безпечно керувати аналізом

Необхідно зазначити, що головні користувачі автоматизованої системи аналізу тексту можуть бути як і фізичні особи так і приватні підприємства, які зацікавленні у певному аналізі тексту для бізнес потреб.

Але цих даних не достатньо і необхідно зробити додатковий аналіз ринкового середовища факторів ризику яка є у таблиці 5.6 та їх можливостей, які надані в таблиці 5.7.

Таблиця 5.6 – Фактори ризику

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Технологічні ризики та проблеми	Будь-які технічні проблеми, які можуть впливати на ефективність роботи чи аналізу тексту	Довгострокова розробка моделі чи самого проєкта, постійне вдосконалення моделі, перенавантаження системи
2	Конкуренція	Поява конкурентів із схожим чи навіть однаковим продуктом на ринкові	Постійне інвестування в рекламу та маркетинг, захист власного продукту
3	Непередбачені ситуації	Будь-які ситуації, які можуть статися по світу над якими ми не маємо контролю	Релокейт бізнесу і усіх його робітників, адекватна реакція на подію та прийняття певних степів компанії для вирішення

Таблиця 5.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Ріст використання штучного інтелекту	Збільшений попит на продукт через зростання використання нейронних мереж	Залучання більшої кількості працівників, розширення сфери впливу
2	Новітні технології та їх інтеграція	Можлива інтеграція із новітніми технологіями для удосконалення алгоритму моделі та ефективності її праці	Інвестування грошей та часу у можливості нових технологій
3	Підвищений попит на аналіз тексту	Зростання попиту на використання схожих технологій для аналізу тексту	Розширення базової функціональності та інвестування у маркетинг існуючого рішення

Після визначення головних факторів ризику, які можуть суттєво підірвати можливість проекту адекватно працювати, або розвиватися. Для цього необхідно провести додатковий аналіз конкуренції на ринку для кращого розуміння, результати якого будуть в таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: Монополістична	В галузі аналізу тексту зазвичай існують численні постачальники з різними підходами та функціоналом.	Потрібно акцентувати увагу на унікальності та перевагах підходу до аналізу тексту.
Рівень конкурентної боротьби: Національний	Конкуренція в цій галузі може бути значущою на національному рівні, з урахуванням широкого застосування аналізу тексту в різних секторах економіки.	Спрямованість на національний ринок вимагатиме адаптації стратегій до місцевих особливостей та потреб клієнтів.
Галузева ознака: Внутрішньогалузева	Компанія конкурує в одній конкретній галузі - аналізу тексту та його застосувань.	Потрібно зосередитися на розвитку глибокого розуміння та експертизи у галузі.
Конкуренція за видами товарів: Товарно-родова	Конкуренція відбувається між різними варіаціями систем аналізу тексту, які можуть мати власні особливості та переваги.	Важливо виділити унікальні риси системи та пояснити, чому вони важливі для клієнтів.
Характер конкурентних переваг: Нецінова	Ключова конкурентна перевага базується на	Спрямованість на нецінові переваги

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
	якості, інноваціях, брендів та інших нецінових факторах.	вимагатиме постійного вдосконалення технологій та зосередження на якісних аспектах продукту.
Інтенсивність конкуренції: Марочна	Будування сильного бренду та позиціонування на ринку як виробника із визначеними унікальними якостями.	Необхідно інвестуватися в маркетинг, розвиток бренду та створення унікального образу на ринку.

Після якісної оцінки, можливо зробити висновок, що тип конкуренцію буде мати монополістичний характер, рівень конкурентів боротьби буде визначатися як національний, галузева ознака буде внутрішньою, а конкуренція буде набувати статусу товарно-родовою.

Після якісної оцінки конкуренції на ринку наступним кроком стане оцінка за моделлю М. Портера.

Головна мета якого буде вивчення більш детальніші за умови конкуренції в галузі. Дані аналізу наведені в таблиці 5.9.

Таблиця 5.9 - Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	TextInspector, QuestionPro	Головною проблемою виходу на ринок є маркетоговий хід та ефективність аналізу тексту.	-	Кількість клієнтів та їх прибутки	Замалі витрати на підтримку власного продукту та найкраще співвідношення ціни та якості
Висновки	Дуже інтенсивна конкуренція	Реальна можливість вийти на ринок навіть за такими умовами	-	Різні вимоги клієнтів впливають на ціни та функціонал і ефективність роботи програми.	Конкуренти можуть бути більш привабливим варіантом через ціну або якість

Після детального аналізу було виявлено високу конкретність на даному ринку. Наступний пункт розгляду буде обґартування факторів конкурентоспроможності у таблиці 5.10

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Новий підхід до рішення проблеми	Встановлення нового алгоритму вирішення проблеми використовуючи модифікований спосіб LSTM мережі
2	Швидкість	За допомоги движка пайтон а також можливості розбиття на пропроцесінгу та загальні легкі шари, в нас не займає багато часу щоб проаналізувати текст, тим самим ми зекономлюємо час для наших юзерів, які не повинні чекати на довгий аналіз
3	Зручність	Під час конкуренції важливу роль відіграє зручність інтерфейсам, яким буде користуватися юзер, оскільки це напряду впливає на ефективність методу та швидкодію між фронтом та беком.

Після аналізу факторів конкурентоспроможності необхідно також визначити рейтинги продуктів конкурентів та власного. Ці результати будуть наведені в таблиці 5.11

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваною системою						
			-3	-2	-1	0	+1	+2	+3
1	Новий підхід до рішення проблеми	15				+			
2	Швидкість	16					+		
3	Зручність	16					+		

Далі необхідно зробити SWOT аналіз. Дані аналізу будуть наведені в таблиці 5.12.

Таблиця 5.12 – SWOT – аналіз стартап-проекту

Сильні сторони: • легкість використання • швидкість аналізу	Слабкі сторони: • замало функціоналу • нема реклами
Можливості: • розвиток технології; • зростання попиту з часом.	Загрози: • дуже велика конкуренція; • відсутність постійних користувачів

Після SWOT аналізу постає питання щодо альтернатив ринкового впровадження стартап-проекта, оскільки стратегія розвитку проекту була обрана на основі поточної спеціалізації. Ця інформація надається в таблиці 5.13.

Таблиця 5.13 - Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Термін реалізації
1	Proof of chain	Низька	4 місяця
2	Презентація Proof of chain	Висока	1 місяць
3	Впровадження Proof of chain у проект та його підтримка	Низька	8 місяців
4	Рекламна компанія	Середня	5 місяців

Найкращим варіантом залученням ресурсів в даному випадку буде презентація Proof of chain як реального додатка, який буде спроможний на самостійну роботу й подальший його розвиток. Також набувається середній шанс на залучення ресурсів через додаткове знайомство під час рекламної компанії.

5.4 Розроблення ринкової стратегії проекту

Будуть розглянуті різноманітні способи та можливості захоплення певного ринка. Першим кроком буде визначення потенційних споживачів, така інформація буде надана в таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Сучасні споживачі	Висока	Середня	Середня	Висока
2	Споживачі яким за 35 років	Середня	Маленька	Маленька	Середня
3	Споживачі до 20 років	Висока	Середня	Середня	Висока
Які цільові групи обрано: сучасні споживачі, Споживачі до 20 років					

Далі буде сформовано єдину базову стратегію розвитку, яку буде надано у таблиці 5.15.

Таблиця 5.15 - Базова стратегія розвитку стартап-проєкту

№	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розробка поточної версії продукту та розгортка його для публічного доступу	Презентувати результати роботи на конференції	Залучити потенціальні групи за допомогою безкоштовних або тріальних підписок	Стратегія спеціалізації

З урахуванням потреб користувачів та тенденції до безкоштовних фіч, написання продовського продукту та надання можливості його тестувати та поліпшувати в рамках розробки – це й є найефективніша стратегія, а точніше стратегія спеціалізації.

Маючи цю стратегію, постає питання визначення стратегії конкурентної поведінки. Ця інформація надається у таблиці 5.16.

Таблиця 5.16. – Визначення базової стратегії конкурентної поведінки.

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Так, з упором на нових споживачів	Так, скоріше за все певна функціональність чи певні фічі для покращення розуміння та інтеракції у користувача	Зайняття конкурентної ніші

Тепер постає необхідність визначити стратегію позиціонування, для того щоб юзери мали змогу ідентифікувати та класифікувати наші продукти, дані яких можна спостерігати в таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Можливість користуванням продукта за умови тріала чи тимчасової підписки, зручний інтерфейс, швидкість відповіді від сервера	Спеціалізація	Різні групи споживачів чи користувачів мають можливість користування продуктом	Якість, швидкість, надійність

В першу чергу будуть наведені ключові переваги концепції потенційного товару. Дані надані в таблиці 5.18.

Таблиця 5.18 - Ключові переваги концепцій потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Потреба у постійному доступі та роботі	Запити юзера будуть завжди надходити на сервер, і сервер завжди буде мати декалька реплік	Доступність
2	Мати локалізацію для різних людей з світу	Можливість обрати власну мову для більш зручного користування	Зручність
3	Захистити енд-поїнти на апі, та надати шифрування	Ваші персональні дані будуть шифруватися, і наш сервіс не буде зберігати ваші дані у відкритому доступі	Безпека

Далі постає питання про опис трьох рівнів моделі майбутнього товару. Дана інформація надана в таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Система автоматичного аналізу тексту із застосуванням нейронної мережі
II. Товар у реальному виконанні	Сервіс який швидко надає потрібний результат в зручному інтерфейсі і надає результати з великою точністю
	Марка: “ТС1”
	Пакування: zip
	Якість: Тести для перевірки надійності та стабільності. Будуть як юніт так і інтеграційні тести, отримання певних сертифікатів
III. Товар із підкріпленням	До продажу: функціональна система
	Після продажу: всі функції системи, підтримка
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності	

Наступним важливим кроком буде встановлення певних меж по цінам на певні групи товарів та їх порівняння із аналогами. Такі дані наведені в таблиці 5.20.

Таблиця 5.20 - Визначення меж встановлення ціни

Рівень цін на товаризамінники	Рівень цін на товари аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Відсутні	Починаючи від 8\$ на місяць в залежності від пакета послуг	Середній	Нижня: 10\$ Верхня: 249\$

Тепер необхідна система збуту товару для реалізації попереднього плану, дані якого наведені в таблиці 5.21.

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Многоразова купівля яка обмежена по часу	Повна технічна підтримка, оновлення софту, доступність сервісів	Необмежена	Комбінована

Останнє що залишається, це визначитися із маркетинговими комунікаціями, дані якого є в таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Фокус на покращення сервісу	Соціальні мережі, інтернет, живе спілкування	Зручність використання, функціональність, якість	Надати цільовим клієнтам інформацію про основні переваги продукту	Практичний показ роботи продукту.

Висновок до розділу 5

Було зроблене заглиблення в складне дослідження стартапу, зосередженого навколо розробки великої автоматизованої системи аналізу тексту з використанням нейронної мережі. Було ретельно досліджено різні сфери застосування цієї послуги, проведено комплексний аналіз конкурентів і визначено технологічні можливості, необхідні для успішного виконання проекту.

Було роздивлено оновлені підходи до розробки, пріоритетом було задоволення потреб користувачів, а також дослідження позиції продукту на ринку через оцінку SWOT-аналізу та системи продажів. Оцінено різні варіанти реалізації продукту на ринку та визначено вагомі переваги концепції проекту.

Щоб сприяти зростанню стартапів, заключна частина аналізу зосереджена на різних аспектах, включаючи три-рівневу модель продукту, маркування, упаковку, контроль якості та встановлення меж комфорту користувача, усе з наміром культивувати сприятливе середовище.

ВИСНОВКИ

Під час написання магістерської дисертації була розроблена система автоматичного аналізу тексту з використанням нейронної мережі. Для демонстрації моделі роботи було використано клієнтський інтерфейс для зручного використання та перевірки значень.

Були проаналізовані існуючі методи обробки тексту, зокрема такі популярні методи, як тематичне моделювання, яке надало два найвідоміші методи тематичного моделювання - Латентну Дирихле Алокацію та Латентний Семантичний Аналіз, завдяки яким відкрилися можливості для автоматичного визначення тем і призначення їх текстам. Також була запущена тема загальних методів класифікації текстового контенту, до яких входять та фільтрація даних пошуку.

Поновлено Бінарне зв'язування разом із методами глибокого навчання, які дійсно мали велику і значну роль у дослідженні та написанні методів обробки тексту. Розібрали головну особливість BR, яке постає у застосуванні багатьох бінарних класифікаторів, кожен з яких може відповідати певній категорії, а також забезпечує високу точність і надійність. А методи глибокого навчання дали нам можливість використовувати рекурентні нейрони мережі.

Оглянули тему послідовних та непослідовних моделей, виявили різницю між ними, які різні типи конструкції і що є особливостями кожного з них, особливо таких, як моделі RNN або LSTM на пару із іншими.

Було обговорено теми програмного забезпечення та технічної підтримки, де визначили основні технічні аспекти нашого проекту, обговорили клієнтський інтерфейс для зручного використання користувачем під назвою Gradio, який має велику кількість методів та функцій для роботи з нейронними мережами, що зробило його найкращим вибором серед усіх. Крім того, були проаналізовані та обговорені різноманітні бібліотеки та їх роль у проекті, без яких нейронна мережа була б неможлива.

Наступним кроком було теоретичне та практичне написання комплексу двосторонньої мережі на основі LSTM, яке показало високі та ефективні результати навчання під час тестування та практичних випробувань. Деталі написання нейромережі були розкриті звітно, і було проведено багато робіт з тестування різних даних. Також було створено та змінено власний набір даних, які ми в подальшому використовували для навчання наших моделей.

В остаточному підсумку було розроблено стартовий проект на основі впровадження системи автоматичного аналізу тексту, під час якого було вирішено багато питань, таких як переваги та недоліки системи, її основні можливості, конкурентні можливості та шляхи поширення серед наявних покупців.

Розроблена система рекомендацій відповідає поставленим цілям. Виконує обов'язковість і за допомогою розробленого проекту аналізу тексту демонструє функціональність та ефективність інструменту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Обробка природної мови (NLP) у Python з кодом (Частина 4. Тематичне моделювання). URL: <https://oleg-dubetcky.medium.com/%D0%BE%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0-%D0%BF%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D0%BE%D1%97-%D0%BC%D0%BE%D0%B2%D0%B8-nlp-%D1%83-python-%D0%B7-%D0%BA%D0%BE%D0%B4%D0%BE%D0%BC-%D1%87%D0%B0%D1%81%D1%82%D0%B8%D0%BD%D0%B0-4-%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B5-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-2df0292f0563>
2. A Beginner's Guide to Latent Dirichlet Allocation(LDA) URL: <https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>
3. Latent Semantic Analysis and its Uses in Natural Language Processing URL: <https://www.analyticsvidhya.com/blog/2021/09/latent-semantic-analysis-and-its-uses-in-natural-language-processing/>
4. Optimal performance of Binary Relevance CNN in targeted multi-label text classification URL: <https://www.sciencedirect.com/science/article/pii/S0950705123010341>
5. ОГЛЯД ПРИНЦИПІВ ГЛИБОКОГО НАВЧАННЯ ЯК ДИНАМІЧНОЇ ТЕОРІЇ ШТУЧНОГО ІНТЕЛЕКТУ URL: https://www.tech.vernadskyjournals.in.ua/journals/2021/6_2021/16.pdf
6. Recurrent Neural Networks (RNN) URL: <https://pub.aimind.so/recurrent-neural-networks-rnn-48c10c443607>
7. Building a Neural Network Zoo From Scratch: The Long Short-Term Memory Network URL: <https://medium.com/mlearning-ai/building-a-neural-network-zoo-from-scratch-the-long-short-term-memory-network-1cec5cf31b7>
8. What is a convolutional neural network (CNN)? URL: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network>
9. Understanding Transformer Neural Network Model in Deep Learning and NLP URL: <https://www.turing.com/kb/brief-introduction-to-transformers-and-their-power>
10. What Is Natural Language Processing (NLP)? URL: <https://www.oracle.com/eg/artificial-intelligence/what-is-natural-language-processing/>
11. LSTM – мережі довгої короткострокової пам'яті URL: <https://habr.com/ru/companies/wunderfund/articles/331310/>

12. Проблема зникання градієнту URL:
<https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0%D0%B7%D0%BD%D0%B8%D0%BA%D0%B0%D0%BD%D0%BD%D1%8F%D0%B3%D1%80%D0%B0%D0%B4%D1%96%D1%94%D0%BD%D1%82%D1%83#:~:text=%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0%20%D0%BF%D0%BE%D0%BB%D1%8F%D0%B3%D0%B0%D1%94%20%D0%B2%20%D1%82%D1%96%D0%BC%2C%20%D1%89%D0%BE,%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%83%20%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%83%20%D0%B2%D1%96%D0%B4%20%D0%BF%D0%BE%D0%B4%D0%B0%D0%BB%D1%8C%D1%88%D0%BE%D0%B3%D0%BE%20%D0%BD%D0%B0%D1%82%D1%80%D0%B5%D0%BD%D0%BE%D0%B2%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>.
13. Understanding Gated Recurrent Unit (GRU) in Deep Learning URL:
<https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2>
14. What is LSTM for Text Classification? URL:
<https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/>
15. Як працюють трансформатори URL: <https://pro.arcgis.com/de/pro-app/3.1/tool-reference/geoai/how-transformers-work.htm>
16. BERT: A Beginner-Friendly Explanation URL:
<https://medium.com/@igniobydigitate/bert-a-beginner-friendly-explanation-876549f0ece2>
17. Understanding Causal LLM's, Masked LLM's, and Seq2Seq: A Guide to Language Model Training Approaches URL:
https://medium.com/@tom_21755/understanding-causal-llms-masked-llm-s-and-seq2seq-a-guide-to-language-model-training-d4457bbd07fa
18. BERT URL: <https://paperswithcode.com/method/bert>
19. A Basic Recurrent Neural Network Model URL:
<https://arxiv.org/pdf/1612.09022.pdf>
20. Understanding Bidirectional RNN in PyTorch URL:
<https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66>
21. Understanding Feed Forward Neural Networks With Maths and Statistics URL: <https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network>
22. Understanding Feedforward Neural Networks URL:
<https://learnopencv.com/understanding-feedforward-neural-networks/>
23. Support Vector Machine — Introduction to Machine Learning Algorithms URL: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
24. SVM(Support Vector Machine) URL:
<http://oryzaindica.com/svm1.html>

25. Гіперплощина в машинному навчанні URL: <https://itwiki.dev/data-science/ml-reference/ml-glossary/hyperplane>
26. Support Vector Machine (SVM) – easily explained URL: <https://databasecamp.de/en/ml/svm-explained>
27. Using Support Vector Machines for Survey Research URL: <https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research>
28. What are Naïve Bayes classifiers? URL: <https://www.ibm.com/topics/naive-bayes>
29. Naive Bayes Introduction URL: <https://www.codingninjas.com/studio/library/naive-bayes-introduction>
30. What Is Python? URL: <https://aloha.co/blog/why-python>
31. Django introduction URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
32. What Is Artificial Intelligence? URL: <https://builtin.com/artificial-intelligence>
33. What is JavaScript? URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
34. The Good and the Bad of .NET Framework Programming URL: <https://www.altexsoft.com/blog/the-good-and-the-bad-of-net-framework-programming/>
35. What is Jupyter Notebook? URL: <https://domino.ai/data-science-dictionary/jupyter-notebook>
36. Ласкаво просимо до Colab! URL: <https://colab.research.google.com/notebooks/welcome.ipynb?hl=uk>
37. What is Microsoft Azure Notebooks? URL: <https://www.devopsschool.com/blog/what-is-microsoft-azure-notebooks-and-use-cases-of-microsoft-azure-notebooks/>
38. Gradio — Build, deploy, and share your Machine Learning Models URL: <https://ismailouahbi.medium.com/gradio-build-deploy-and-share-your-machine-learning-models-7b38baba659c>
39. Introduction to TensorFlow URL: <https://www.tensorflow.org/learn>
40. What is PyTorch? URL: <https://www.nvidia.com/en-us/glossary/pytorch/>
41. pip (система керування пакунками) URL: [https://uk.wikipedia.org/wiki/Pip_\(%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D0%B0%D0%BA%D1%83%D0%BD%D0%BA%D0%B0%D0%BC%D0%B8\)](https://uk.wikipedia.org/wiki/Pip_(%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D0%B0%D0%BA%D1%83%D0%BD%D0%BA%D0%B0%D0%BC%D0%B8))
42. What is a kaggle? - <https://www.kaggle.com/discussions/general/328265>
43. Vectorization Techniques in NLP [Guide] URL: <https://neptune.ai/blog/vectorization-techniques-in-nlp-guide>

44. 7 types of Layers you need to know in Deep Learning and how to use them URL: <https://inside-machinelearning.com/en/7-types-of-layers-you-need-to-know-in-deep-learning-and-how-to-use-them/>

45. What is Epoch in Machine Learning? URL: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning>

46. What are Binary Cross-Entropy and Log Loss? URL: [https://www.aporia.com/learn/understanding-binary-cross-entropy-and-log-loss-for-effective-model-monitoring/#:~:text=Binary%20Cross%20Entropy%2FLog%20Loss,*%20log\(1%20%2D%20p\)\)](https://www.aporia.com/learn/understanding-binary-cross-entropy-and-log-loss-for-effective-model-monitoring/#:~:text=Binary%20Cross%20Entropy%2FLog%20Loss,*%20log(1%20%2D%20p)))