Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Вебсервіс рекомендацій «YTSimilar» для інтелектуального пошуку споріднених відео на YouTube
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5

Єгор Вілянський
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект

(повна назва освітньої програми)
Керівник ст. викл. Олена Гриньова
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Вілянському Єгору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Вебсервіс рекомендацій «YTSimilar» для інтелектуального пошуку споріднених відео на Youtube

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.
3. Вихідні дані до роботи Youtube, Python, HTML, CSS, JavaScript, Visual Studio 2022, Google Cloud, Youtube API, Flask, yt-dlp, youtube-trainscript-api

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Характеристика інструментів та технологій розробки
3) Розробка програмного забезпечення

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05-22.05	виконано
3	Постановка задачі	23.05-25.05	виконано
4	Обґрунтування вибору використання технологій	26.05-29.05	виконано
5	Програмна реалізація	30.05-04.06	виконано
6	Оформлення пояснювальної записки	05.06-13.06	виконано
7	Підготовка до захисту	14.06-17.06	виконано
8	Захист перед ЕК	18.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Олена Гриньова
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 62 с., 14 рис., 1 дод., 16 джерел.

МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ПРИРОДНОЇ МОВИ, ПОШУК ВІДЕО, РЕКОМЕНДАЦІЙНА СИСТЕМА, FLASK, GOOGLE CLOUD, NLP.

Кваліфікаційна робота присвячена розробці вебсервісу YTSimilar, що виконує семантичний пошук споріднених відео на платформі YouTube.

Метою проєкту є побудова інтелектуальної рекомендаційної системи, що використовує сучасні технології машинного навчання та обробки природної мови. У процесі реалізації було здійснено аналіз наукових публікацій та протестовано моделі обчислення тематичної подібності між відео. Для реалізації сервісу застосовано стек технологій такі як Python, Flask, Google Cloud, yt-dlp, youtube-transcript-api та бібліотеки векторизації тексту.

Сервіс базується на клієнт-серверній архітектурі, де клієнтська частина реалізована у вигляді браузерного розширення, що взаємодіє з сервером для аналізу введених посилань. Система отримує транскрипт відео, виконує попередню обробку тексту, порівнює семантичний зміст, та формує список рекомендованих відео. В результаті реалізовано прототип, який продемонстрував релевантність рекомендацій навіть у випадках, коли вбудовані алгоритми YouTube не здатні забезпечити точного тематичного збігу.

Розроблений вебсервіс може стати основою для подальших досліджень у галузі персоналізованих рекомендацій, освітнього контенту та аналітики користувацької поведінки.

ABSTRACT

Bachelor’s thesis contains: 62 pp., 14 fig., 1 ann., 16 references.

FLASK, GOOGLE CLOUD, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, NEURAL NETWORKS, NLP, RECOMMENDATION SYSTEM, VIDEO SEARCH.

The qualification thesis is dedicated to the development of the YTSimilar web service, which performs semantic search for related videos on the YouTube platform.

The objective of the project is to build an intelligent recommendation system that employs modern machine learning technologies and natural language processing. During the implementation process, relevant academic publications were analyzed and models for computing thematic similarity between videos were tested. The service is implemented using a technology stack that includes Python, Flask, Google Cloud, yt-dlp, youtube-transcript-api, and text vectorization libraries. The system is based on a client-server architecture, where the client side is realized as a browser extension that interacts with the server to analyze submitted video links. The system retrieves the video transcript, performs preprocessing of the text, compares semantic content, and generates a list of recommended videos.

As a result, a prototype was developed that demonstrated the relevance of its recommendations even in cases where YouTube’s built-in algorithms failed to provide accurate thematic matches. The developed web service may serve as a foundation for further research in the fields of personalized recommendations, educational content discovery, and user behavior analytics.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання;

СКБД – система керування базами даних;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

ASR – Automatic Speech Recognition – автоматичне розпізнавання мовлення;

CD – Continuous Deployment – автоматичне розгортання.

CI – Continuous Integration – безперервна інтеграція;

CNN – Convolutional Neural Networks – згорткові нейронні мережі;

CSS – Cascading Style Sheets – мова стилів;

HTML – Hypertext Markup Language – мова розмітки;

JS – JavaScript – мова для створення інтерактивних та динамічних веб-сторінок;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

LSTM – Long Short-Term Memory – короткочастна пам’ять;

ML – Machine Learning – машинне навчання;

MVP – Minimum Viable Product – мінімально життєздатний продукт;

NLP – Natural Language Processing – обробка природної мови;

RNN – Recurrent Neural Networks – рекурентна нейронна мережа;

SVM – Support Vector Machine – опорний вектор.

ВСТУП

У сучасну цифрову епоху обсяг інформації, що щоденно створюється та споживається в мережі, стрімко зростає. Особливе місце в цьому інформаційному середовищі займає відеоконтент, який є ефективним інструментом навчання, комунікації, самореалізації та розваг. Відеоплатформа YouTube стала однією з ключових у глобальній інфраструктурі розповсюдження мультимедійної інформації, об’єднуючи мільярди користувачів і пропонуючи доступ до мільйонів відео з різноманітною тематикою.

Попри розвинену систему рекомендацій, що базується на методах штучного інтелекту, платформа YouTube не завжди забезпечує релевантний пошук для вузькоспеціалізованих або нових відео. Це зумовлює потребу в розробці альтернативних підходів до формування рекомендацій, здатних забезпечити більш глибокий аналіз вмісту відео, зокрема текстової складової – транскриптів.

У межах кваліфікаційної роботи було розроблено інтелектуальну систему YTSimilar, яка виконує пошук схожих відео на основі семантичного аналізу транскриптів і метаданих. Система використовує сучасні методи обробки природної мови (Natural Language Processing), машинного навчання та векторизації тексту для визначення тематичної подібності між відеоматеріалами.

Розробка базується на клієнт-серверній архітектурі. Клієнт реалізовано у вигляді розширення для браузера на основі Chromium таких як Google Chrome та Opera, що взаємодіє з вебсервісом, побудованим за допомогою фреймворку Flask. Серверна частина відповідає за отримання транскриптів, попередню обробку тексту, вибір релевантних відео за допомогою YouTube API, а також обчислення рівня подібності між відео на основі TF-IDF і косинусної міри.

Метою кваліфікаційної роботи стало проектування та реалізація методів інтелектуального пошуку схожих відео на платформі YouTube шляхом аналізу транскриптів і метаданих.

Технологічна основа базується на використанні таких інструментів та техологій, як Python (основна мова програмування), Flask (локальний сервер), youtube-transcript-api, yt-dlp, scikit-learn, Levenshtein, HTML/CSS/JavaScrip, а також YouTube Data API. Запропонований підхід продемонстрував свою ефективність у знаходженні релевантних відео за змістом, навіть коли алгоритми YouTube не видають таких результатів.

Окрему увагу в рамках практики приділено теоретичному обґрунтуванню обраних технологій, аналізу предметної галузі, виявленню обмежень сучасних платформ та формуванню логіки функціонування кожного модуля системи. Запропоноване рішення дозволяє поєднати інтелектуальні обчислення з високою продуктивністю, будучи цінним як кінцевому користувачу, так і слугувати за основу для подальшої науково-прикладної розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Вступ до предметної галузі

Упродовж останнього десятиліття відеохостинг YouTube перетворився на одну з провідних платформ цифрового контенту у світі. Щодня користувачі переглядають мільярди відео, охоплюючи широкий спектр тем – від освітніх матеріалів до розважальних шоу. Разом з активним зростанням обсягів доступного відеоконтенту зростає потреба у вдосконалених інструментах навігації, які здатні ефективно фільтрувати, класифікувати та рекомендувати відео, релевантні інтересам користувача.

Традиційні системи рекомендацій, що використовуються на платформі YouTube, базуються на алгоритмах колаборативної фільтрації, персоналізації та аналізу поведінкових даних. Проте ці підходи мають низку обмежень, зокрема схильність до ефекту «інформаційної бульбашки», недостатню гнучкість у пошуку за змістом і неврахування семантичної подібності між відео. Як результат, користувачі можуть втратити доступ до менш популярних, але тематично близьких матеріалів.

У сфері інформаційного пошуку та інтелектуального аналізу даних все більше уваги приділяється використанню методів машинного навчання, обробки природної мови (NLP) [1] та векторного представлення текстової інформації для побудови рекомендаційних систем нового покоління. Такі системи орієнтовані не лише на аналіз історії користувача, а й на розуміння змісту відео шляхом обробки його метаданих [2] – назви, опису, тегів тощо [3].

У цьому контексті розробка вебсервісу «YTSimilar» набуває особливої актуальності. Його основна функція полягає у формуванні списку споріднених відео на основі семантичної подібності, що дозволяє покращити точність рекомендацій та забезпечити більш осмислений і персоналізований пошук. Застосування таких підходів відкриває нові

можливості для розвитку інтелектуальних систем підтримки прийняття рішень у цифровому середовищі та підвищення якості користувацького досвіду.

1.2 Актуальність

Актуальність розробки вебсервісу рекомендацій «YTSimilar» обумовлена стрімким зростанням обсягів відеоконтенту на платформі YouTube, що ускладнює процес ефективного пошуку релевантної інформації. Сучасні користувачі стикаються з проблемою надлишку контенту, внаслідок чого значна частина цінної та тематично близької інформації залишається поза їх увагою. У таких умовах традиційні алгоритми рекомендацій часто не забезпечують достатнього рівня точності [4], оскільки орієнтуються переважно на популярність відео, історію переглядів та поведінкові патерни, ігноруючи глибинну семантику відеоматеріалів.

У світлі розвитку інструментів обробки природної мови, технологій машинного навчання та семантичного аналізу виникає потреба в нових підходах до побудови рекомендаційних систем. Зокрема, дедалі більшої ваги набуває концепція семантично обґрунтованих рекомендацій, що передбачає пошук спорідненого контенту не за формальними ознаками, а за змістовною подібністю. Саме такий підхід покладено в основу вебсервісу «YTSimilar», який використовує методи векторного представлення текстових описів відео для виявлення релевантних матеріалів.

Зважаючи на високий попит на інтелектуальні системи рекомендацій як серед пересічних користувачів, так і в професійних сферах (освіта, медіааналітика, маркетинг тощо), впровадження такого сервісу є своєчасним та суспільно значущим. YTSimilar сприяє не лише покращенню якості користувацького досвіду, а й розширює можливості доступу до

якісного та тематично релевантного відеоконтенту, що в умовах інформаційного перевантаження має особливу цінність.

1.3 Існуючі системи рекомендацій YouTube та їхніх обмежень

Система рекомендацій YouTube є однією з найпотужніших у сучасному цифровому середовищі. Вона базується на складних алгоритмах машинного навчання, які обробляють величезні обсяги даних, зокрема історію переглядів користувача, тривалість взаємодії з контентом, оцінки, коментарі, підписки, а також враховують популярність відео і темпи зростання переглядів. У межах системи можна виокремити кілька ключових механізмів – домашня сторінка формує добірки відео на основі інтересів користувача, список «Up next» який динамічно оновлює пропозиції відповідно до поточного перегляду, результати пошуку, що враховують ключові слова та поведінкові метрики, а тематичні плейлисти пропонують контент за категоріями, що цікавлять користувача.

Водночас, попри значну ефективність у залученні уваги аудиторії, існуюча система рекомендацій має низку обмежень. Її функціонування значною мірою залежить від уже сформованої історії переглядів, що ускладнює пошук нового чи нетипового контенту. Це, у свою чергу, призводить до виникнення ефекту інформаційної бульбашки, коли користувач бачить лише подібні типи відео, що знижує різноманіття доступного контенту. Крім того, система здебільшого спирається на зовнішні сигнали, як-от кількість переглядів чи вподобань, і не завжди враховує глибинну семантику змісту відео, що може спричинити появу лише поверхнево дотичних рекомендацій. Ще однією проблемою є домінування популярного контенту: алгоритми підсилюють відео з високими показниками взаємодії, часто ігноруючи менш відомі, але потенційно корисні матеріали.

1.4 Методи машинного навчання та штучного інтелекту в побудові рекомендаційних систем

Сучасні рекомендаційні системи значною мірою базуються на досягненнях у галузі машинного навчання (ML) та штучного інтелекту (AI), що дозволяє ефективно обробляти великі масиви даних, виявляти приховані закономірності та формувати персоналізовані пропозиції для користувачів. Завдяки швидкому розвитку алгоритмів, рекомендаційні механізми стали здатними враховувати не лише явні інтереси користувачів, але й контекст взаємодії, семантичний зміст інформації та динаміку змін у поведінці.

Використання ML та AI у рекомендаційних системах включає як класичні підходи – наприклад, колаборативну та контентну фільтрацію, – так і сучасні глибокі нейронні мережі, які здатні моделювати складні залежності між об’єктами та користувачами. Особливої популярності набувають трансформерні архітектури, системи на основі векторного представлення (embedding) та моделі обробки природної мови, що дозволяють формувати рекомендації, спираючись на глибинний аналіз описів, назв, тегів та інших метаданих.

У межах розробки вебсервісу «YTSimilar» методи машинного навчання використовуються для побудови системи семантичного пошуку споріднених відео. Це передбачає перетворення текстових атрибутів відео в числові вектори, розрахунок їх подібності та формування ранжованого списку релевантних результатів. Такий підхід забезпечує новий рівень якості рекомендацій, що базуються не лише на популярності чи історії переглядів, а на глибокому розумінні змісту відеоматеріалів.

Штучний інтелект та машинне навчання – є концепціями, які існують вже протягом кількох десятиліть і були створені з метою надання комп’ютерам можливості виконувати завдання, які традиційно виконує людина. До таких завдань належать, зокрема, виявлення спаму, класифікація об’єктів та інші типові види когнітивної діяльності.

У більш вузькому розумінні машинне навчання можна охарактеризувати як галузь штучного інтелекту, що досліджує та розробляє методи і алгоритми, здатні до самонавчання. Ці методи дозволяють системам самостійно вдосконалюватися на основі аналізу вхідних даних, виявляючи закономірності в інформаційних масивах з метою оптимізації процесів і прийняття рішень.

В рамках інтелектуального аналізу даних вхідною інформацією для побудови моделей машинного навчання виступають експериментальні дані, представлені у вигляді структурованих множин вимірювань, що забезпечує можливість їх подальшого оброблення та навчання моделей (1.1).

$$X(n, m), \tag{1.1}$$

де n – кількість вимірів;

m – кількість характеристик, що вимірювались.

Кожен акт виміру характеристик певного явища можна розглядати як об’єкт, що знаходиться в m-мірному просторі ознак. Отже, можемо ввести таке поняття як матриця об’єктів та ознак (1.2).

$$X = \left|\left|a_j(x_i)\right|\right|, \tag{1.2}$$

де $a_j: \rightarrow D_j$;

$j=\overline{1,\ldots m}$ – ознаки об’єктів.

Зазначимо, що при:

– $D_j = \{0,1\}$ – ознака a_j є бінарною;

– $|D_j| < \infty$ – ознака a_j є номінальною;

– $|D_j| < \infty$, D_j – впорядкована множина, ознака a_j є порядковою;

– $D_j = R$ – ознака a_j є кількісною.

Набір експериментальних даних є реалізацією деякого процесу (1.3).

$$f: X \to Y, \tag{1.3}$$

де X – набір вхідних даних;

Y – вектор цільових значень заданого процесу.

Оскільки під час проведення експериментів можливі похибки, що можуть спричинити неповноту або спотворення даних, у процесі навчання моделей часто використовують ймовірнісний підхід. Для формалізації цього процесу вводиться функція щільності розподілу p(x,y), яка описує невідомий імовірнісний розподіл на множині X×Y. У такому випадку залежність f можна інтерпретувати як імовірнісний розподіл, що визначає зв'язок між вхідними даними та відповідями (1.4).

$$p(x, y) = p(x) \times p\left(\frac{y}{x}\right), \tag{1.4}$$

де p(y/x) = δ(y - f(x)), δ(x) – δ-функція;

f(x) – значення цільової змінної.

Таким чином, можемо задати модель сумісної щільності розподілу об'єктів навчальної вибірки та значень цільової функції, що буде апроксимувати невідому щільність (1.5).

$$p(x, y) - \phi(x, y, \theta). \tag{1.5}$$

Позначимо через θ параметр моделі, за якого експериментальна вибірка є найбільш ймовірною, тобто максимально узгоджується з моделлю щільності. У випадку, якщо всі спостереження у вибірці X є незалежними, тоді спільну щільність розподілу для всіх елементів вибірки можна записати у такому вигляді (1.6).

$$p(X) = p\big((x_1, y_1), \dots (x_n, y_n)\big) = p(x_1, y_1) \times \dots \times p(x_n, y_n), \quad (1.6)$$

де n – кількість об’єктів в експериментальній вибірці.

Звідси, можна отримати функцію правдоподібності (1.7).

$$L(\theta, X) = \Pi_{i=1}^{n} \varphi(x_i, y_i, \theta), \quad (1.7)$$

де φ (x, y, θ) – модель сумісної щільності розподілу;

n – кількість об’єктів в експериментальній вибірці;

θ – параметр моделі.

Чим більше значення функції правдоподібності, тим краще модель відповідає експериментальній вибірці. Відповідно до методу максимальної правдоподібності, процес навчання полягає в максимізації цієї функції з метою отримання найточнішої оцінки параметра θ, тобто визначення оптимальної моделі для заданої сумісної щільності розподілу. Таким чином, завдання навчання моделі зводиться до розв’язання задачі оптимізації.

У загальному випадку, як зазначалося раніше, модель машинного навчання будується на основі апроксимації сумісного ймовірнісного розподілу (1.8).

$$p(x, y), \quad (1.8)$$

де x – деякий об’єкт навчальної вибірки;

y – значення цільової функції на даному об’єкті.

Моделі, які апроксимують даний розподіл, називаються генеративними. Однак, на практиці досить часто характеристичний простір ознак об’єкта x є дуже високої розмірності, що значно ускладнює побудову генеративних моделей. Через таку особливість, дослідники при вирішенні практичних задач зазвичай будують дискримінантні моделі, тобто такі, що моделюють умовну ймовірність (1.9).

$$p\left(\frac{x}{y}\right), \tag{1.9}$$

де x – деякий об’єкт навчальної вибірки;

y – значення цільової функції на даному об’єкті.

Отже, генеративна модель намагається змоделювати повний розподіл даних, щоб мати змогу апроксимувати значення цільової функції для конкретного об’єкта. Натомість дискримінативна модель зосереджується лише на наближенні значення цільової функції для певного прикладу, не враховуючи повний розподіл вхідних даних.

Більшість сучасних алгоритмів машинного навчання належать саме до дискримінативного типу. Машинне навчання, як одна з провідних галузей ШІ, охоплює широкий спектр методів та алгоритмів, вибір яких залежить від специфіки вхідних даних, типу навчання та вимог до подальшого застосування моделі.

Загалом методи машинного навчання прийнято поділяти на три основні категорії:

– методи з вчителем (Supervised Learning);

– методи без вчителя (Unsupervised Learning);

– методи з підкріпленням (Reinforcement Learning).

Основна мета методів з учителем полягає у відтворенні значення цільової змінної на основі наданих характеристик об’єкта (рисунок 1.1) [5].

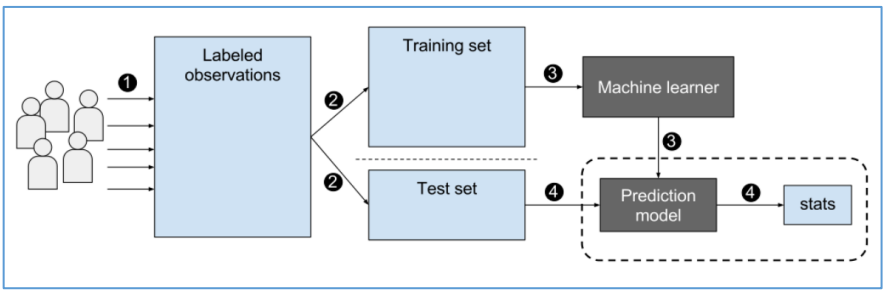

Рисунок 1.1 – Схема архітектури навчання та використання моделі машинного навчання з учителем

Далі розглянемо кожен з цих напрямів докладніше.

Методи з учителем активно застосовуються в багатьох практичних задачах, зокрема в розпізнаванні мови, виявленні об’єктів на зображеннях, класифікації текстів, оцінці кредитоспроможності позичальників та прогнозуванні фінансових показників, таких як ціни акцій.

Схема ілюструє стандартний процес навчання та застосування моделі машинного навчання в режимі навчання з учителем. Вона включає послідовність етапів, починаючи з підготовки даних і закінчуючи оцінкою ефективності побудованої моделі.

Початковим етапом є наявність розмічених даних (Labeled Observations), які складаються з набору прикладів, де кожен елемент містить вхідні ознаки (features) та відповідні цільові значення (мітки). Наприклад, у задачі класифікації зображень такими даними можуть бути фотографії разом із підписами, що визначають їхній клас.

Наступним кроком є розділення даних на навчальний та тестовий набори (Training set / Test set). Навчальний набір використовується для тренування моделі, тоді як тестовий набір залишається незалежним і призначений для оцінки якості роботи моделі після завершення навчання. Таке розділення є критично важливим для забезпечення об’єктивності оцінювання.

Далі алгоритм машинного навчання (Machine Learner) аналізує навчальний набір і будує прогностичну модель (Prediction Model). В залежності від поставленої задачі можуть використовуватися різні методи, такі як лінійна регресія, дерева рішень, методи опорних векторів або нейронні мережі. Метою цього етапу є виявлення закономірностей між вхідними даними та мітками, що дозволяє моделі робити передбачення на нових даних. Послідовність процесу можна представити у вигляді ланцюга (рисунок 1.2).

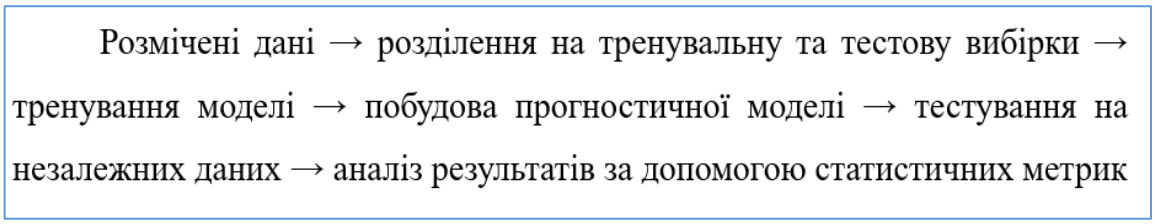

Рисунок 1.2 – Послідовність процесу

Після навчання модель проходить оцінку (Test set та stats) на тестовому наборі даних. Для аналізу її ефективності застосовуються різні статистичні метрики, такі як точність (accuracy), F1-середнє, середньоквадратична помилка (Mean Squared Error, MSE) тощо. Ці показники дозволяють визначити, наскільки добре модель узагальнює знання, отримані під час тренування, і чи є її прогнози достовірними.

2 ХАРАКТЕРИСТИКА ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Огляд API YouTube Data та алгоритмів ранжування YouTube

YouTube API є важливим інструментом для розробки інтелектуальних систем рекомендацій, оскільки дозволяє отримувати доступ до великих обсягів даних, що зберігаються на платформі. Завдяки API, розробники можуть інтегрувати функціональність YouTube в свої додатки, аналізувати контент, взаємодії користувачів і створювати персоналізовані рекомендації. API YouTube стає основою для побудови інтелектуальних вебсервісів, які використовують машинне навчання та обробку даних для підвищення точності рекомендацій та покращення користувацького досвіду.

YouTube API забезпечує доступ до великого обсягу різноманітних даних, зокрема інформації про відео, коментарі, підписки, плейлисти та інші метадані, що є основою для створення ефективних систем рекомендацій. Завдяки API можна здійснювати запити до платформи, отримувати необхідні дані і інтегрувати їх у сторонні програми, що дозволяє використовувати їх для персоналізації рекомендацій та покращення пошукових систем.

YouTube Data API надає набір інструментів для доступу до даних, що зберігаються на платформі. Це API дозволяє розробникам витягувати, переглядати та взаємодіяти з контентом YouTube, включаючи відео, коментарі, підписки, статистику переглядів, плейлисти та інші метадані [6]. Це потужний інструмент для збору даних, який підтримує різноманітні запити і дає змогу користуватися інформацією в режимі реального часу.

YouTube Data API забезпечує широкі можливості для взаємодії з контентом платформи, зокрема надає доступ до метаданих відео, включаючи тривалість, опис, кількість переглядів, реакції, дату публікації, а також дозволяє працювати з коментарями, що відкриває можливості для

аналізу користувацької взаємодії. Крім цього, API дає змогу отримувати дані про підписки користувачів, що дозволяє формувати уявлення про їхні інтереси, а також витягувати інформацію про наявні плейлисти, які можуть бути використані для побудови персоналізованих рекомендацій. Окрему категорію становить доступ до додаткових метаданих, таких як теги, категорії чи авторство, що розширює аналітичний потенціал інструменту.

Формування запитів до YouTube Data API здійснюється із застосуванням різноманітних параметрів, які дозволяють точково отримувати необхідну інформацію. Зокрема, ідентифікатори відео або каналу використовуються для доступу до конкретних ресурсів, пошукові запити – для фільтрації відео за ключовими словами або тегами, параметри обмеження кількості результатів – для контролю обсягу відповіді, а категоріальні фільтри – для звуження вибірки відповідно до тематики, наприклад, музика, освіта чи спорт.

YouTube Data API має певні обмеження, які необхідно враховувати при роботі з великими обсягами даних. Одним із основних лімітів є кількість запитів, які можуть бути здійснені за певний період часу. Кожен запит до API має певний ліміт ресурсів, і з кожним запитом цей ліміт витрачається. Це означає, що з часом для здійснення великих запитів може знадобитися більше часу або ресурсів.

Таким чином, YouTube Data API є потужним інструментом для отримання доступу до даних на платформі YouTube, але його ефективне використання потребує врахування обмежень і лімітів, що можуть впливати на процес розробки та впровадження рекомендаційних систем.

Алгоритми ранжування відео на YouTube формують порядок відображення результатів пошуку та пропозицій у стрічці рекомендацій на основі складної комбінації сигналів, індексації та моделювання поведінки користувачів [7]. Головною метою цих алгоритмів є максимізація часу перегляду (watch time) і загального рівня залучення користувачів, оскільки тривалі перегляди є надійним індикатором якості відео [8]. Крім того, лайки,

дизлайки та коментарі розглядаються як сигнали соціального підтвердження, що сприяють підвищенню релевантності та популярності відео. Метадані відео – заголовки, описи та теги – також відіграють важливу роль, оскільки правильно оптимізовані ключові слова допомагають алгоритму ідентифікувати тематику контенту та покращують його видимість у відповідних запитах. Алгоритми аналізують історію переглядів і взаємодій конкретного користувача, щоб сформувати індивідуальний профіль уподобань і запропонувати максимально релевантний контент. Крім того, система ранжування враховує показники популярності відео, такі як загальна кількість переглядів і швидкість їхнього приросту (velocity), що дозволяє алгоритму надавати пріоритет відео з високим темпом зростання аудиторії [9].

Для персоналізації рекомендацій YouTube застосовує кілька методів машинного навчання. Колаборативна фільтрація аналізує схожість уподобань між користувачами або між відео на основі історії переглядів і вподобань, використовуючи як user-based, так і item-based підходи. Контентна фільтрація, натомість, ґрунтується на характеристиках самого відео – метаданих і транскриптах – щоб знайти матеріали, які є тематично суміжними з тими, що вже переглядав користувач. Крім цих традиційних методів, YouTube активно впроваджує глибокі нейронні мережі, здатні моделювати більш складні залежності між характеристиками контенту та поведінковими патернами користувачів, що забезпечує підвищену точність рекомендацій при обробці великих обсягів даних [10].

2.2 Використання алгоритмів машинного навчання для рекомендаційних систем

У сучасному машинному навчанні нейронні мережі, безумовно, відіграють важливу роль і виступають універсальним інструментом для вирішення широкого кола задач. Водночас слід підкреслити, що вони не є

єдиним можливим підходом. У багатьох випадках традиційні методи машинного навчання демонструють високу ефективність, зокрема при роботі зі складними наборами даних, при цьому маючи перевагу в простоті реалізації, інтерпретованості результатів і меншій обчислювальній складності. У подальших підрозділах буде розглянуто як сучасні методи на основі нейронних мереж, так і перевірені часом класичні алгоритми, які залишаються конкурентоспроможними в багатьох практичних задачах [11].

2.2.1 Алгоритм Random Forest

Одним із найпоширеніших класичних підходів, що належать до ансамблевих методів машинного навчання, є алгоритм Random Forest (випадкового лісу). Цей метод особливо популярний у задачах класифікації, проте також активно застосовується у задачах регресії. Його основними перевагами є простота реалізації, висока швидкість навчання та здатність ефективно працювати з великими обсягами даних.

Метод випадкового лісу (Random Forest) полягає в побудові ансамблю дерев рішень. Кожне дерево в лісі формує власне передбачення, а остаточний результат визначається за принципом більшості голосів у випадку класифікації або середнього значення в задачах регресії. Ідея алгоритму ґрунтується на концепції «мудрості натовпу» – об’єднання великої кількості слабких моделей у вигляді дерев дозволяє отримати більш стійке та точне передбачення, ніж у випадку використання окремої моделі.

Кожне дерево в ансамблі Random Forest створюється на основі випадкової підвибірки з навчального набору даних (метод бутстрепінгу). У процесі побудови дерева, на кожному його вузлі випадковим чином вибирається підмножина змінних, серед яких визначається найкраще можливе розбиття. Це зменшує кореляцію між окремими деревами та сприяє кращому узагальненню моделі. Зазвичай дерева не підлягають

обрізанню після побудови й ростуть до максимальної глибини на своїх підвибірках, що дозволяє їм максимально охоплювати структуру даних.

Серед основних переваг алгоритму слід назвати його універсальність. Він підходить як для задач класифікації, так і для регресійного аналізу. Крім того, метод ефективно працює з різними типами змінних, зокрема як категоріальними, так і числовими. Йому не потрібна попередня нормалізація або стандартизація ознак, що спрощує підготовку даних. Важливо й те, що Random Forest залишається стійким до пропущених значень у вибірці, зберігаючи при цьому високу точність.

Водночас, використання цього підходу супроводжується низкою недоліків. Через велику кількість дерев, що формуються у процесі навчання, алгоритм вимагає значних обчислювальних ресурсів. Це також впливає на швидкість навчання. Воно може бути досить повільним при роботі з великими обсягами даних. Окрім того, модель складно інтерпретувати. Визначення вагомості окремих змінних у фінальному рішенні ускладнюється через взаємодію між великою кількістю дерев.

Попри всі ці недоліки, метод Random Forest залишається одним із найбільш ефективних та надійних класичних підходів до моделювання. Його реалізації доступні у численних програмних бібліотеках, зокрема в середовищах R та Python, що забезпечує широке використання цього алгоритму в дослідницьких і прикладних завданнях.

2.2.2 Логістична регресія

Одним із базових та найпоширеніших методів машинного навчання, що активно застосовується для розв’язання задач класифікації, є логістична регресія. Цей підхід дозволяє встановити залежність між вхідними ознаками та ймовірністю належності об’єкта до певного класу. Обчислена ймовірність, у свою чергу, використовується для прийняття остаточного рішення про класифікацію.

У випадку, коли маємо справу з двома класами (наприклад, класифікація повідомлень як спам/не спам), застосовується бінарна логістична регресія. Якщо ж кількість класів перевищує два, використовується її узагальнення – мультиноміальна (поліноміальна) логістична регресія.

Логістична регресія належить до класу лінійних моделей, основна мета яких полягає у побудові гіперплощини, що розділяє простір ознак, забезпечуючи таким чином розмежування об'єктів різних класів. Найпростіша форма цієї моделі в задачах бінарної класифікації – це лінійна комбінація вхідних ознак, до якої застосовується сигмоїдна функція, що перетворює її на значення в діапазоні від 0 до 1, інтерпретоване як ймовірність належності до одного з класів. Найпростіша лінійна модель бінарного класифікатора має вигляд (2.1).

$$y(x) = sign\left(\sum_{i=1}^{n}(w_i \times x_i) = sign(\overrightarrow{w^T}\,\vec{x}\right). \qquad (2.1)$$

У логістичній регресії передбачається, що об'єкт належить до певного класу на основі знаку лінійної комбінації його ознак та вагових коефіцієнтів його моделі. Таким чином, процес навчання полягає в підборі оптимальних значень цих вагових коефіцієнтів, що найкраще відповідають наявним навчальним даним.

На відміну від звичайної лінійної регресії, логістична модель не просто визначає клас, до якого належить об'єкт, а обчислює ймовірність належності цього об'єкта до певної категорії. Це ключова відмінність, яка дозволяє застосовувати логістичну регресію саме в задачах класифікації.

Основною гіпотезою моделі є припущення, що ймовірність належності об'єкта до конкретного класу повинна бути обмежена інтервалом від 0 до 1. Оскільки результат лінійної моделі може набувати значень поза межами цього діапазону, необхідно використати спеціальну

нелінійну трансформацію. У логістичній регресії таку роль виконує сигмоїдна функція – одна з базових функцій у машинному навчанні, яка відображає будь-яке дійсне значення у відрізок [0; 1] Вона і забезпечує коректне відображення виходу моделі у вигляді ймовірності. Формула даної функції виглядає наступним чином (2.2).

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.2}$$

Графік сигмоїдної функції зображено на рисунку 2.1.

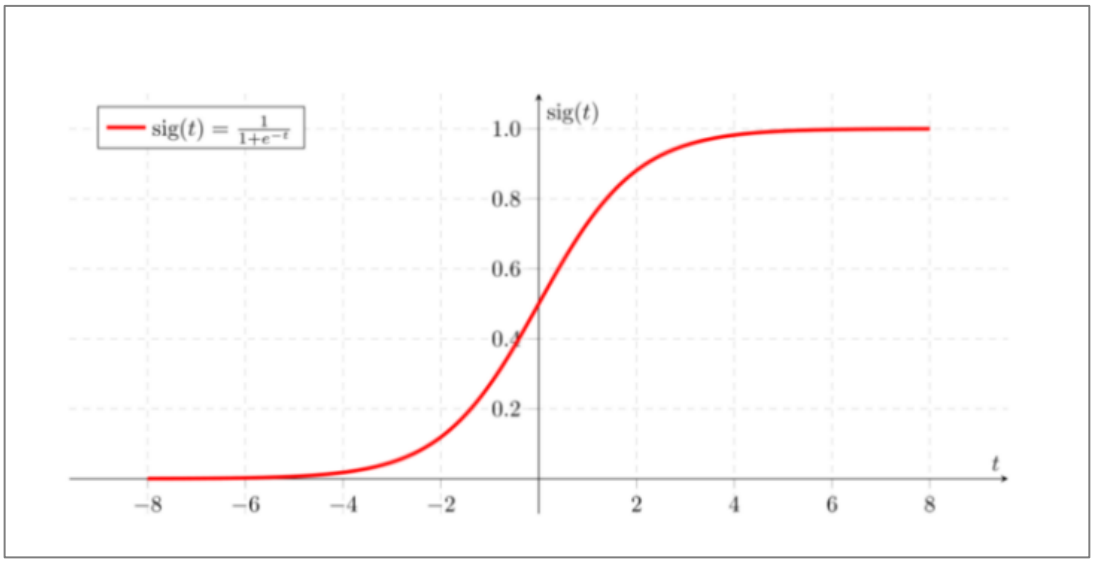

Рисунок 2.1 – Графік сигмоїдної функції

Логістична регресія належить до класу базових методів машинного навчання та широко застосовується для розв’язання задач класифікації. Її популярність зумовлена відносною простотою реалізації, доступністю інтерпретації результатів та ефективністю в умовах обмежених обчислювальних ресурсів. Завдяки ймовірнісному підходу, модель дозволяє не лише визначати належність об’єкта до певного класу, але й оцінювати ступінь впевненості в прийнятому рішенні. Важливою перевагою

логістичної регресії є її здатність масштабуватись до багатокласових задач, що забезпечується шляхом використання поліноміальних розширень. Модель характеризується високою швидкістю обробки нових даних і порівняно низькою схильністю до перенавчання, однак при наявності великої кількості ознак доцільним є застосування методів регуляризації.

Разом із тим, логістична регресія має низку обмежень, які знижують її ефективність у складних практичних застосуваннях. Зокрема, модель передбачає наявність лінійного зв’язку між незалежними змінними та логарифмічними коефіцієнтами, що значно ускладнює її застосування в умовах, коли вхідні дані мають складну нелінійну структуру. У таких випадках логістична регресія поступається за точністю більш сучасним підходам, зокрема нейронним мережам, методам градієнтного бустингу та іншим ансамблевим алгоритмам, які здатні моделювати складні залежності між ознаками.

2.2.3 Метод опорних векторів (Support Vector Machines)

Серед поширених методів машинного навчання не менш популярним, ніж розглянуті раніше алгоритми, є метод опорних векторів (Support Vector Machine, SVM). Цей алгоритм належить до класу методів з навчанням під наглядом і застосовується як для задач класифікації, так і для регресійного аналізу.

Основна концепція SVM полягає у побудові гіперплощини, яка розділяє простір ознак таким чином, щоб забезпечити якомога чіткіше розмежування між класами. У випадку двовимірного простору ця гіперплощина представлена у вигляді прямої, яка поділяє площину на дві частини, де кожна відповідає певному класу. Метою алгоритму є побудова такої гіперплощини, яка максимізує відстань до найближчих точок обох класів – це дозволяє покращити узагальнюючу здатність моделі.

Об'єкти, що розміщені найближче до роздільної гіперплощини, називаються опорними векторами. Саме вони визначають положення та орієнтацію гіперплощини, і відіграють ключову роль у навчанні моделі.

Метод SVM показує високу ефективність не лише у бінарній класифікації. Його можна адаптувати до задач із більш ніж двома класами шляхом зміни ядра, що буде детальніше розглянуто у подальших розділах.

Для кращого розуміння принципу роботи алгоритму у випадку бінарної класифікації розглянемо вектор нормалі $\vec{w}$, який є перпендикулярним до гіперплощини. Припустимо, що об'єкти позитивного класу позначаються як x_2, а негативного – як x_1. Необхідно визначити проєкцію вектора, що з'єднує опорні вектори обох класів, на вектор $\vec{w}$. Ця проєкція визначає ширину роздільної смуги між класами – саме цей показник підлягає максимізації під час навчання моделі.

Отримана геометрична інтерпретація дозволяє перейти до формального математичного опису алгоритму (2.3).

$$\langle (x_+ - x_-), \frac{\vec{w}}{||\vec{w}||} \rangle = \frac{1}{||\vec{w}||} \times (\langle x_+, \vec{w} \rangle - \langle x_-, \vec{w} \rangle) = \frac{2}{||\vec{w}||} \to max \to |||\vec{w}| \to|$$

$$\to min \leftrightarrow \frac{\left(\overrightarrow{w^T}\vec{w}\right)}{2} \to min. \quad (2.3)$$

На рисунку 2.2. можна побачити принцип роботи алгоритму.

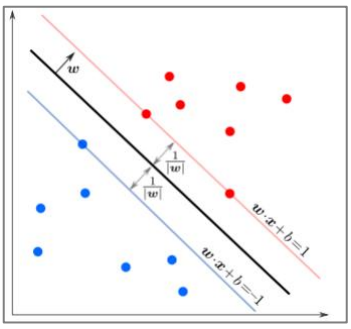

Рисунок 2.2 – Принцип побудови максимальної роздільної полос

Метод опорних векторів (SVM) є розширенням лінійного класифікатора, яке дозволяє вирішувати нелінійні задачі класифікації шляхом трансформації вихідного простору ознак у простір вищої розмірності. Ця трансформація здійснюється з використанням ядрових функцій, які забезпечують можливість побудови роздільних поверхонь довільної форми. Завдяки такому підходу, навіть якщо початкові дані не є лінійно розділеними, після перетворення вони можуть стати такими, що істотно розширює область застосування методу.

Ядрова функція відіграє визначальну роль у забезпеченні ефективності методу опорних векторів, оскільки саме вона визначає спосіб трансформації простору ознак і характер взаємодії між об’єктами даних. Універсальність SVM проявляється в його здатності демонструвати стабільно високі результати навіть за відсутності повної інформації про природу вхідних даних. Метод ефективно застосовується для аналізу неструктурованих і напівструктурованих даних, включаючи тексти, зображення та складні ієрархічні структури. За умови правильного вибору ядра, алгоритм здатен вирішувати задачі підвищеної складності, забезпечуючи при цьому низький рівень перенавчання, що є важливою перевагою у порівнянні з альтернативними підходами. Крім того, метод добре масштабується до наборів даних різного обсягу, що робить його придатним для широкого спектру практичних застосувань.

Разом з тим, SVM має певні обмеження, що варто враховувати при його використанні. Зокрема, вибір ядрової функції є нетривіальним завданням, яке вимагає емпіричного підходу та ретельного налаштування. У випадку обробки великих обсягів даних навчання моделі може потребувати значних обчислювальних ресурсів. Крім того, складність внутрішньої структури SVM, а також обмежена прозорість прийнятих ним рішень, створюють труднощі в інтерпретації результатів і їх подальшій адаптації до специфічних вимог прикладних доменів.

SVM є надзвичайно корисним інструментом для первинного аналізу набору даних, особливо коли невідомо, яка модель найкраще підходить. Це зумовлює його широку популярність у прикладних задачах. Реалізації методу доступні у багатьох програмних середовищах, зокрема в мовах програмування Python та R, де представлені спеціалізовані бібліотеки для роботи з SVM.

2.2.4 Наївний Баєс (Naive Bayes)

Серед найпоширеніших алгоритмів класифікації особливе місце посідає наївний баєсівський класифікатор. Його назва зумовлена двома ключовими аспектами. По-перше, алгоритм ґрунтується на наївному припущенні, згідно з яким усі ознаки (функції) є незалежними одна від одної. Це припущення рідко виконується в реальних умовах, однак дозволяє суттєво спростити розрахунки. По-друге, математичну основу алгоритму становить теорема Байєса, на честь якої класифікатор і отримав відповідну частину своєї назви.

Далі розглянемо формальне представлення теореми Байєса, яка лежить в основі цього методу (2.4).

$$P(d) = \frac{P(c) \times P(c)}{P(d)}. \tag{2.4}$$

Ймовірність того, що об’єкт d належить до класу ccc, позначається як P(c|d). Тут P(d|c) – це ймовірність появи об'єкта d за умови, що він належить до класу ccc; P(c) – апріорна ймовірність появи класу ccc; а P(d) – безумовна ймовірність появи об'єкта d серед усієї сукупності даних.

Алгоритм наївного баєсівського класифікатора функціонує за наступною схемою. На першому етапі формується матриця частот, яка відображає кількість появ ознак у різних класах. На її основі обчислюється

матриця правдоподібності, що характеризує ймовірності для кожної ознаки з урахуванням класу. Далі, використовуючи формулу теореми Байєса, визначається апріорна ймовірність для кожного класу, яка слугує основою для передбачення.

Наївний баєсівський класифікатор, хоча й базується на відносно простій теоретичній основі, демонструє ефективність у широкому спектрі прикладних задач, особливо у багатокласових сценаріях. Його продуктивність значною мірою залежить від структури навчальних даних – якщо вибірка є збалансованою, модель забезпечує стабільно високу точність, тоді як при наявності диспропорцій між класами або упередженості даних точність може суттєво погіршуватись.

Однією з ключових переваг цього методу є висока швидкість навчання і класифікації, що робить його придатним для задач, де критично важливим є час обробки. У випадках, коли виконується припущення незалежності ознак, алгоритм здатен перевершувати за точністю складніші моделі при значно нижчих вимогах до обсягу навчальних даних. Крім того, цей підхід демонструє високу ефективність при роботі з категоріальними змінними.

Водночас модель має ряд обмежень. Проблема нульової частоти – коли тестова вибірка містить категорії, які були відсутні в навчальних даних і це призводить до того, що відповідна ймовірність дорівнює нулю, унеможливлюючи класифікацію. Також припущення про незалежність ознак на якому базується модель у більшості реальних випадків не виконується, що може негативно впливати на точність. Незважаючи на ці недоліки, наївний баєсівський класифікатор залишається дієвим базовим інструментом у побудові статистичних моделей і широко використовується в практиці завдяки доступності реалізацій у таких мовах, як Python та R.

2.2.5 Згорткові нейронні мережі

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) вперше привернули до себе широку увагу у 2012 році, коли модель, розроблена Алексом Крижевським, здобула перемогу в престижному конкурсі ImageNet, знизивши рівень помилок класифікації з 26% до 15%. Відтоді нейронні мережі активно застосовуються в різних галузях, зокрема такими IT-гігантами, як Facebook, Google та Amazon, які використовують їх для підвищення ефективності власних систем.

Згорткові нейронні мережі належать до класу алгоритмів машинного навчання, що найчастіше застосовуються для розв’язання задач класифікації зображень. Водночас, CNN також демонструють високу ефективність у сфері обробки текстів. Основу їхньої функціональності становлять дві ключові операції: операція згортки, яка дозволяє виділяти локальні ознаки, та операція об'єднання (pooling), що сприяє зменшенню розмірності даних при збереженні найважливішої інформації.

Архітектура згорткових нейронних мереж є багаторівневою, тобто вихідні дані одного шару передаються на вхід наступного. Це забезпечує ефективне поетапне виявлення ознак, необхідних для точної класифікації. Один із прикладів такої архітектури представлено на рисунку 2.3.

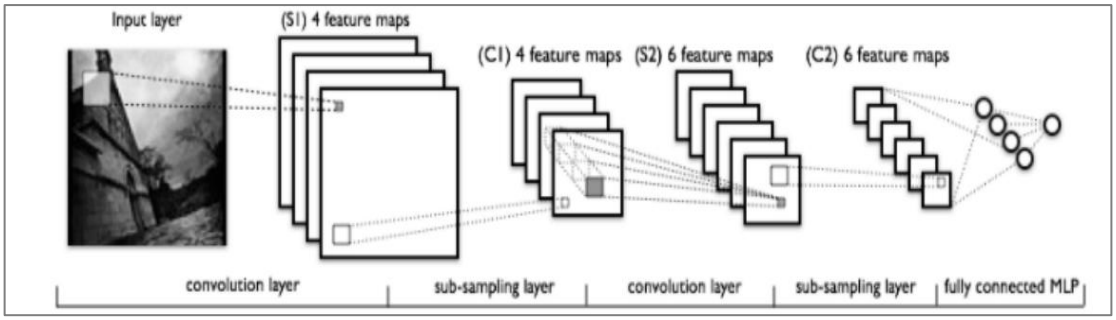

Рисунок 2.3 – Приклад архітектури багатошарової згорткової нейронної мережі

Розглянемо докладніше принцип роботи операцій згортки та об’єднання у згорткових нейронних мережах.

Операція згортки. Її реалізація залежить від типу вхідних даних, які використовуються під час тренування або тестування моделі. Хоча згорткові нейронні мережі найчастіше застосовуються для класифікації зображень, вони також успішно використовуються для обробки текстової інформації, тому операція згортки адаптується відповідно до характеру даних. Розглянемо особливості згортки як для зображень, так і для тексту.

Згортка при обробці зображень. У комп’ютерному представленні зображення задається у вигляді числової матриці, де кожне значення знаходиться в межах від 0 до 255 і визначає яскравість кольору. Існує багато колірних просторів, проте одним з найпоширеніших є RGB (Red, Green, Blue), у якому кожне кольорове зображення представляється трьома окремими матрицями яскравості для кожного з основних кольорів [12]. Для спрощення розгляду подальшого матеріалу, зосередимось на обробці чорно-білого зображення, яке задається однією матрицею яскравості [7].

Першим кроком операції згортки є визначення фільтра згортки (або ядра), який буде застосовано до вхідної матриці зображення. Фільтр накладається на матрицю таким чином, щоб його верхній лівий кут співпадав із відповідним елементом зображення. Далі виконується поелементне множення значень фільтра і значень пікселів під ним, після чого обчислюється сума добутків, яка записується до нової матриці як перший елемент.

Після цього фільтр зміщується праворуч на фіксовану кількість кроків (величину цього зсуву задає параметр stride), і операція повторюється. Коли фільтр доходить до кінця рядка, він зміщується вниз і продовжує обробку наступного рядка зображення. Цей процес триває доти, доки не буде охоплено всю вихідну матрицю. Отримана в результаті згортки матриця називається картою ознак (feature map).

Операція згортки може виконуватися паралельно з використанням кількох фільтрів. У такому випадку результатом буде набір вихідних матриць (карт ознак), які далі можуть бути використані для наступних етапів обробки.

Візуалізацію процесу виконання операції згортки представлено на рисунку 2.4.

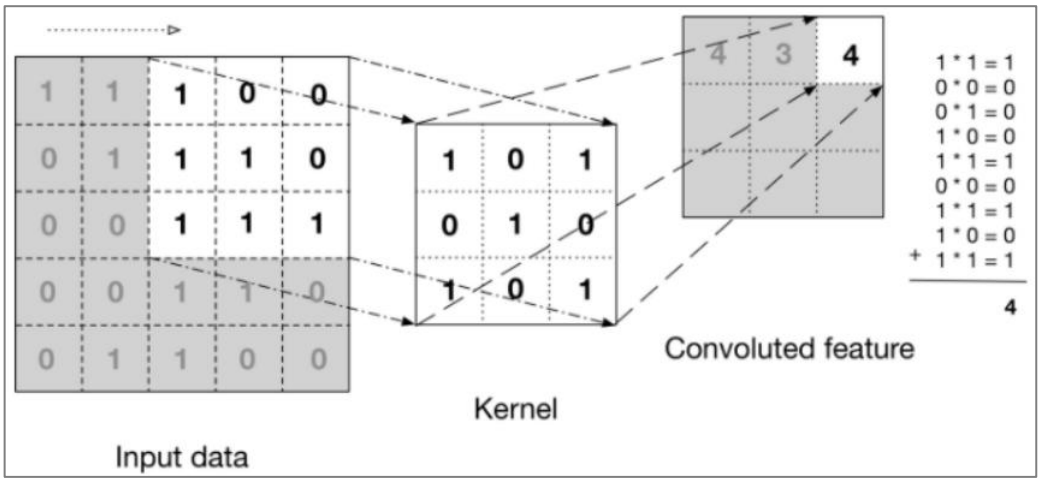

Рисунок 2.4 – Приклад виконання операції згортки

Розглянемо особливості функціонування згортки у випадку роботи з текстовими даними. Припустимо, що маємо послідовність із nnn слів, кожне з яких представлене вектором розмірності d. У цьому випадку застосовується одномірне згорткове ядро шириною hhh, яке виконує скалярний добуток між конкатенованими векторами в межах згорткового вікна та ваговими коефіцієнтами ядра. Після виконання цієї операції зазвичай застосовується нелінійна функція активації.

Як зазначалося вище, будь-яке речення можна подати у вигляді матриці розмірності n×d, де nnn – кількість слів, а d – розмірність векторного представлення кожного слова. Аналогічно до обробки кольорових зображень, для текстових даних можна використовувати кілька фільтрів, а

результуючі значення можуть бути об'єднані шляхом підсумовування або конкатенації отриманих матриць.

Наступним етапом після згортки є операція об’єднання (англ. pooling), яка виконується з метою зниження розмірності (дискретизації) вхідних даних. Принцип її роботи полягає у застосуванні певної математичної функції до локальних фрагментів карти ознак, подібно до операції згортки, але без використання вагових коефіцієнтів. Замість цього, у межах кожного локального регіону обирається певне значення – найчастіше це максимум(max pooling) або середнє арифметичне (average pooling).

Зокрема, під час виконання операції об’єднання вибирається вікно певного розміру, яке послідовно переміщується по всій матриці ознак. У межах цього вікна виконується відповідна функція – обчислення максимуму, середнього значення або іншої метрики. Результатом є нова матриця меншого розміру, яка відображає узагальнені характеристики вхідних даних. Приклад роботи цього процесу наведено на рисунку 2.5.

Метою використання pooling-шару є поступове зменшення просторових розмірів карт ознак, що дозволяє моделі зосередитися на найбільш інформативних ділянках вхідного зображення або тексту, а також зменшити обчислювальну складність наступних шарів.

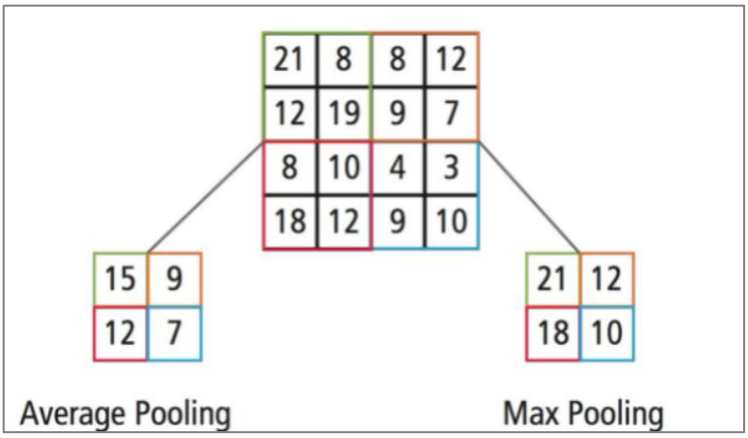

Рисунок 2.5 – Приклад виконання операції об’єднання для згорткових нейронних мереж

Операція об’єднання для текстових даних за своєю структурою не відрізняється від аналогічної операції для зображень. В результаті формується вихідний вектор, який відображає ключові характеристики та ознаки кожного конкретного речення.

Зазвичай останніми шарами згорткових нейронних мереж є повністю зв’язані персептронні шари. Оскільки операції згортки та об’єднання виконують функцію екстракції ознак, саме на основі отриманих ознак навчається нейронна мережа, яка відповідає за класифікацію об’єктів.

Для побудови такого типу мереж існує велика кількість бібліотек і модулів для Python, що пояснюється широким поширенням та популярністю згорткових мереж у сучасних задачах класифікації.

2.2.6 Рекурентні нейронні мережі (RNN)

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) є одними з перших сучасних алгоритмів, здатних зберігати інформацію про попередні вхідні дані. Основна ідея рекурентних мереж полягає у роботі з послідовністю інформації. На відміну від традиційних нейронних мереж, які припускають незалежність усіх вхідних та вихідних даних, RNN враховують зв’язок між ними. Наприклад, у задачі перекладу для правильного відтворення послідовності слів необхідно, щоб модель зберігала інформацію про кожне раніше згенероване слово. Назва «рекурентні» пояснюється тим, що мережі виконують однакові операції для кожного елемента послідовності, а результат залежить від попередніх обчислень, адже мережа має «пам’ять», що зберігає інформацію про вже оброблені дані [13]. Приклад структури рекурентної нейронної мережі наведено на рисунку 2.6.

Моделі з довготривалою короткочасною пам’яттю (LSTM, Long Short-Term Memory) розроблені для подолання проблеми затухаючого градієнта, що часто виникає в стандартних рекурентних нейронних мережах. Щоб

краще зрозуміти принцип їхньої роботи, варто детальніше розглянути структуру блоку A, зображеного на рисунку 2.6, коли в його основі використовується LSTM-архітектура.

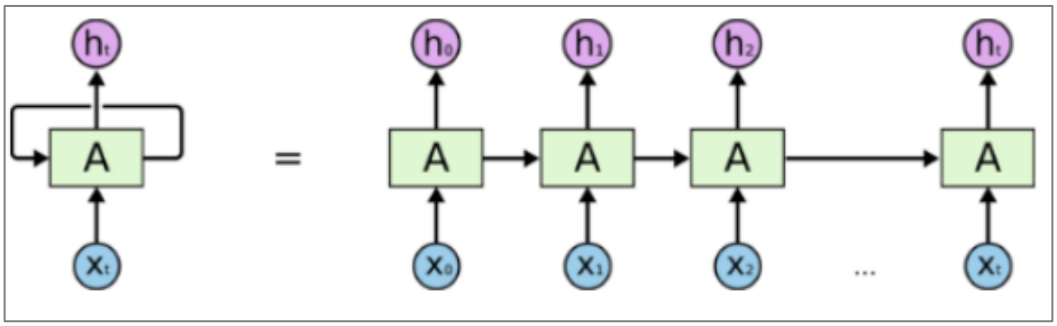

Рисунок 2.6 – Приклад структури рекурентної нейронної мережі

Особливістю LSTM є здатність зберігати інформацію на довгі часові проміжки – ця функціональність є типовою для таких моделей. Їх обчислювальний процес умовно ділиться на три основні компоненти: Forget gate (ворота забування), Input gate (ворота входу) та Output gate (ворота виходу). Кожен із цих компонентів виконує окрему роль у регулюванні потоку інформації крізь комірку пам’яті. Схематичне розташування цих елементів у межах обчислювального блоку наведено на рисунку 2.7.

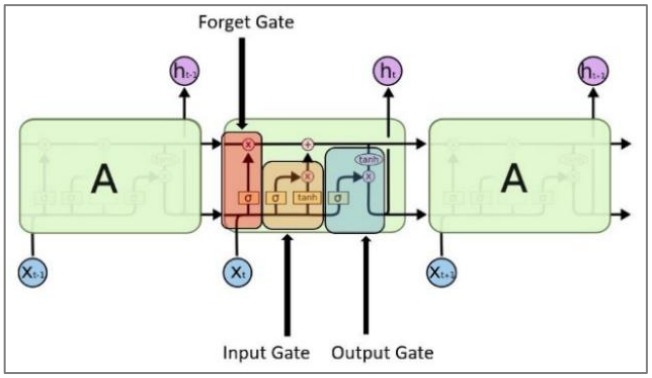

Рисунок 2.7 – Схематичне розміщення обчислювальних модулів LSTM моделі

Розглянемо докладніше, як саме функціонують окремі етапи обчислень у LSTM-моделі та чому при цьому не виникає проблема затухаючого градієнта. Першим елементом, що обробляє вхідну інформацію, є так звані ворота забування (Forget gate). Саме вони відповідають за визначення того, яку частину інформації з попереднього кроку слід зберегти, а яку – відкинути. Кожен з елементів алгоритму, що бере участь в обчисленнях, називають шлюзом або воротами, і надалі ми будемо користуватися цими термінами.

Forget gate приймає рішення про те, що необхідно «забути» на поточному часовому кроці. Це досягається шляхом використання сигмоїдної активаційної функції, яка на вхід отримує попередній прихований стан h_{t-1} і поточний вхід x_t. Результатом роботи цієї функції є значення у діапазоні від 0 до 1, що визначає ступінь збереження інформації в клітинному стані C_{t-1}: 1 означає повне збереження, тоді як 0 – повне забування (2.5).

$$f_t = sigmoid\big(w_f \times [h_{t-1}, x_t] + b_f\big). \quad (2.5)$$

Рекурентні нейронні мережі, такі як моделі LSTM – є ключовими інструментами обробки послідовних даних, що мають тимчасову або логічну залежність між елементами. Їхня здатність зберігати інформацію про попередні кроки значно підвищує ефективність моделювання таких задач, як машинний переклад, розпізнавання мови та аналіз часових рядів. На відміну від класичних RNN, архітектура LSTM дозволяє уникнути проблеми затухаючого градієнта завдяки впровадженню спеціалізованих обчислювальних елементів – воріт забування, входу та виходу. Ці елементи забезпечують контрольований потік інформації, що дозволяє зберігати або відкидувати дані на основі їх значущості, тим самим забезпечуючи стійке навчання моделей навіть на довгих послідовностях. Отже, LSTM-моделі є

важливим кроком у розвитку глибинного навчання для роботи з послідовною інформацією.

2.3 Технології збору, обробки та аналізу даних відео

2.3.1 Обробка метаданих відео

Метадані – це інформація, яка характеризує інші дані, тобто є своєрідними «даними про дані» [14]. Вони можуть включати описові, адміністративні або структурні елементи. Такий підхід до визначення метаданих застосовується фахівцями з управління даними вже протягом багатьох десятиліть. Водночас, метадані відображають атрибути, властивості та теги, що дозволяють описувати і класифікувати інформацію. Точніше буде сказати, що метадані – це відомості, які дають контекст самим даним. Вони можуть містити різноманітні характеристики, що стосуються важливої інформації, зокрема типу даних, автора, дати створення, статусу виконання або способу використання інформації в межах організації. Після визначення метадані надають даним зміст і функціональне призначення, сприяючи їхньому швидкому пошуку, що є необхідною умовою для аналізу великих обсягів даних і побудови звітності.

Крім того, метадані здатні ідентифікувати так звані «малі дані», які відіграють важливу роль у формуванні структури майбутніх великих масивів інформації. У дослідженні [15], опублікованому Harvard Business Review, окреслено три основні відмінності між великими і малими даними:

– великі дані орієнтовані на досягнення організаційних цілей, тоді як малі дані допомагають окремим особам досягати особистих цілей;

– індивідуальні користувачі, як правило, не мають прямого доступу до великих даних.

Великі дані зазвичай контролюються організаціями, у той час як малі дані перебувають під контролем окремих осіб. Організації можуть надавати

людям доступ до великих даних, тоді як користувачі, навпаки, дозволяють компаніям використовувати свої малі дані.

У контексті функціонування сучасних рекомендаційних систем важливою передумовою підвищення точності персоналізованих пропозицій для користувачів є ефективна обробка метаданих відеоконтенту. До таких метаданих належать, зокрема, субтитри відео (заздалегідь завантажені у текстовому форматі), назви відео та пов’язані з ними теги. Обробка зазначених елементів здійснюється у декілька послідовних етапів, кожен з яких має окреме функціональне призначення та спрямований на формування семантичного профілю відео для подальшої оцінки його релевантності інтересам конкретного користувача.

Першим етапом є збір метаданих. До переліку використовуваних джерел належать – субтитри, які можуть бути як автоматично згенерованими, так і створеними вручну автором контенту; назва відео, що у більшості випадків узагальнено відображає його основну тему або зміст; теги, які автор додає з метою тематичної класифікації матеріалу [16]. Ці елементи слугують вихідним матеріалом для подальшого текстового аналізу.

Наступним кроком є попередня обробка текстових даних. Цей процес передбачає очищення вхідної інформації від шумів (зайвих символів, HTML-тегів, пунктуації), перетворення тексту до єдиного формату (зокрема, переведення до нижнього регістру), а також застосування процедур лематизації або стемінгу з метою нормалізації словоформ. Окрім того, з тексту усуваються стоп-слова, які не несуть змістового навантаження, наприклад, службові частини мови.

На основі підготовленого текстового масиву здійснюється аналіз змісту (content analysis). Застосовуються методи побудови векторного представлення тексту, зокрема TF-IDF, Word2Vec, BERT та інші моделі, що дозволяють кількісно оцінити семантичну схожість між різними відео. Субтитри аналізуються з метою виявлення ключових понять, тем, об’єктів

або подій, що у сукупності формують змістовий профіль відео. Назви та теги використовуються для уточнення контексту, категоризації та інтерпретації ймовірної цільової аудиторії.

Після цього відбувається інтеграція отриманих даних із поведінковою інформацією про користувача. Зокрема, враховуються історія переглядів, вподобання, тривалість перегляду окремих відео тощо. На основі цієї інформації формується індивідуалізований профіль інтересів користувача, що дає змогу підвищити точність подальших рекомендацій.

Заключним етапом є побудова списку релевантних відео для користувача. Рекомендаційна система зіставляє векторні подання переглянутих відео з аналогічними векторами інших відеоматеріалів, що є доступними на платформі, і пропонує ті з них, які мають найвищий ступінь семантичної відповідності. Таким чином, використання субтитрів, назв та тегів відео як джерел метаданих дозволяє не лише здійснювати ефективну класифікацію контенту, але й забезпечувати високий рівень персоналізації рекомендацій, що є ключовим завданням для алгоритмів рекомендаційних систем у цифровому середовищі.

Послідовність процесу обробки метаданих відеоконтенту для рекомендаційної системи схематично продемонстрований на рисунку 2.8.

Рисунок 2.8 – Процес обробки метаданих відеоконтенту для рекомендаційної системи

Ефективна обробка метаданих відеоконтенту є необхідною умовою для підвищення точності та релевантності персоналізованих рекомендацій у сучасних цифрових платформах. Метадані, такі як субтитри, назви відео та теги, забезпечують структурований контекст для аналізу змісту відеоматеріалів і дозволяють формувати змістовий профіль кожного об’єкта контенту. Їхня попередня обробка, зокрема очищення, нормалізація та лінгвістичний аналіз, є критично важливою для подальшого застосування методів векторного подання текстів, таких як TF-IDF, Word2Vec або BERT. У поєднанні з поведінковими даними користувача, метадані слугують основою для створення адаптивного механізму рекомендацій, що здатен ефективно задовольняти індивідуальні інформаційні запити. Таким чином, якісна та структурована обробка метаданих забезпечує не лише підвищення продуктивності рекомендаційної системи, але й сприяє зростанню рівня задоволеності користувачів платформою в цілому.

2.3.2 Аналіз змісту відео (теги, опис, транскрипція)

У сучасних інформаційних системах аналіз тегів і опису відео є важливим елементом обробки контенту, спрямованим на підвищення точності класифікації, пошуку та персоналізованих рекомендацій. Теги, як ключові слова або короткі фрази, покликані коротко описувати тематику відео. Вони створюються переважно автором контенту і відображають основні аспекти змісту, зокрема жанр, головні об'єкти, події, цільову аудиторію тощо. Основна функція тегів полягає у полегшенні навігації та фільтрації великого обсягу відеоконтенту, що є особливо актуальним для платформ з високою щільністю користувацького трафіку.

Теги широко використовуються алгоритмами машинного навчання для автоматичної класифікації контенту за тематичними категоріями, формування рекомендаційних списків, а також забезпечення релевантності результатів пошуку. З технічного погляду, обробка тегів передбачає низку

етапів – очищення тексту, нормалізацію, а також семантичне узагальнення. Найбільш поширеними методами є лематизація, що зводить слова до їхньої базової форми, а також фільтрація синонімів з метою уникнення дублювання змісту. Наприклад, для тегів «освіта», «навчання» та «учіння» може бути створена єдина категорія, що відповідає освітній тематиці. Таке групування значно підвищує точність кластеризації та релевантність подальших аналітичних висновків. На платформах, як-от YouTube або Vimeo, теги відіграють ключову роль у формуванні добірок контенту, що відповідає інтересам користувача на основі раніше переглянутих відео або пошукових запитів.

Паралельно з тегами, важливою текстовою складовою є опис відео, який також надається автором і містить більш розгорнуту інформацію про зміст. Типова структура опису охоплює короткий огляд відео, ключові моменти сюжету, заклики до дії (наприклад, підписка або коментарі), а також зовнішні посилання на додаткові ресурси. Змістовний аналіз опису дає змогу здійснити більш точне позиціонування відео в межах тематики, а також оцінити його потенційну привабливість для конкретних аудиторій.

Оцінка релевантності опису до фактичного змісту відео здійснюється за допомогою методів семантичного аналізу. До найпоширеніших підходів належать тематична класифікація, виявлення ключових понять, аналіз контексту та частотності термінів. Такі методи дозволяють сформувати векторне представлення тексту, яке може бути використане у багатовимірних просторах для вимірювання схожості між різними відео. Крім того, автоматизоване зіставлення описів з іншими текстовими елементами (наприклад, транскрипцією або тегами) сприяє створенню єдиного семантичного профілю відео, що значно підвищує точність рекомендаційних систем та сприяє адаптації контенту до індивідуальних уподобань користувача.

Таким чином, аналіз тегів і опису відео є невіддільною частиною інтелектуальної обробки мультимедійної інформації, що забезпечує як

ефективну класифікацію контенту, так і високий рівень персоналізації інформаційних послуг.

Транскрипція відео, тобто текстова розшифровка усного мовлення, становить важливе джерело структурованої інформації, що дозволяє перетворити неформалізований аудіовізуальний контент у формат, придатний для подальшого аналізу. В умовах стрімкого зростання обсягів відеоданих транскрипція набуває особливого значення, оскільки відкриває можливості для використання лінгвістичних та семантичних методів обробки тексту з метою покращення класифікації, тематичного групування та індексації контенту. Існує два основні способи отримання транскрипцій – ручне створення, що передбачає участь людини у розшифруванні мовлення, та автоматичне розпізнавання мовлення (ASR, Automatic Speech Recognition), яке здійснюється із застосуванням спеціалізованих програмних рішень.

Серед сучасних технологій автоматичної транскрипції особливу популярність мають сервіси Google Speech-to-Text, Microsoft Azure Speech та відкриті моделі на кшталт OpenAI Whisper, які демонструють високу точність розпізнавання навіть за умов наявності фонових шумів або різних акцентів. Такі системи здатні не лише перетворювати мовлення на текст, а й розпізнавати мову, інтонації, а в деяких випадках – емоційний фон виступу. Завдяки цьому транскрипція може бути використана як основа для виявлення ключових слів, визначення основної тематики відео, сегментації за змістовими блоками, а також оцінки емоційної тональності мовлення – що, у свою чергу, дозволяє враховувати настрій і стиль подачі інформації під час формування рекомендацій або пошукової видачі.

Застосування транскрипції у рекомендаційних системах забезпечує низку переваг, серед яких – підвищення точності відповідності між змістом відео та запитами користувачів, можливість глибшої семантичної обробки та розширення функціоналу класифікаційних алгоритмів. Наприклад, порівняння транскрипцій різних відео за тематичними або стилістичними

критеріями дозволяє формувати кластери контенту для цільових груп користувачів.

Подальшим кроком у контексті аналізу змісту відео є інтеграція результатів, отриманих з аналізу тегів, опису та транскрипції, у єдиний семантичний профіль. Такий підхід передбачає об’єднання всіх текстових характеристик відео в уніфіковане векторне представлення, що відображає основні змістовні аспекти, лексико-семантичні зв’язки та контекстуальні особливості. Ці вектори можуть бути використані у системах кластеризації, що дозволяє групувати відео за спільними тематиками, стилями або призначенням.

У результаті формується багатовимірна модель, придатна для ефективного застосування в рекомендаційних системах, пошукових алгоритмах та платформах автоматичної класифікації, де глибина та точність розуміння змісту відео відіграють ключову роль для забезпечення релевантного і персоналізованого досвіду користувача.

2.3.3 Визначення тематичної подібності

У сучасних інформаційних системах, орієнтованих на обробку відеоконтенту, надзвичайно важливим завданням є визначення тематичної подібності між окремими одиницями контенту. Цей процес дозволяє виявити спільні змістові характеристики відео, що, у свою чергу, є основою для кластеризації, формування персоналізованих рекомендацій, тематичної індексації та покращення релевантності результатів пошуку. Тематична подібність визначається шляхом порівняння семантичних векторів, побудованих на основі аналізу текстових і метаданих – таких як теги, описи, транскрипції, із застосуванням сучасних алгоритмів обробки природної мови (NLP) та методів векторизації (TF-IDF, word2vec, BERT тощо). У цьому підрозділі буде розглянуто основні підходи до вимірювання

тематичної подібності, а також особливості їхнього використання в контексті аналізу відео з відкритих платформ.

Тематична подібність є ключовим поняттям у сфері обробки інформації, зокрема при аналізі мультимедійного контенту, такого як відео. Вона передбачає оцінку ступеня схожості між об'єктами (відео, документами, повідомленнями) на основі їхнього тематичного наповнення, що дозволяє визначити, наскільки близькі вони за змістом. Тематична подібність широко застосовується у фільтрації контенту, класифікації даних, а також у рекомендаційних системах, де необхідно ідентифікувати матеріали, що відповідають інтересам користувача. Варто розрізняти тематичну та семантичну подібність – перша ґрунтується переважно на спільності тем або ключових слів, тоді як друга враховує глибші контекстуальні та лексичні зв'язки між поняттями.

Для визначення тематичної подібності необхідним є представлення вмісту відео у вигляді векторів у відповідному багатовимірному просторі ознак. Одиницею аналізу можуть виступати як повні відеофайли, так і їх окремі сегменти або фрейми, залежно від завдань обробки. Векторизація ґрунтується на текстових джерелах, пов'язаних із відео – тегах, описах і транскрипціях. Перед побудовою векторів здійснюється попередня обробка даних, що включає очищення тексту від шумів, лематизацію, видалення стоп-слів та інші прийоми нормалізації лексичних одиниць, що сприяють підвищенню якості тематичного моделювання.

Серед методів векторного представлення інформації вирізняються кілька основних підходів. Статистичні моделі, такі як терм-фреквенсі (TF) та TF-IDF (Term Frequency-Inverse Document Frequency), забезпечують базовий рівень узагальнення тексту за допомогою частотного аналізу термінів у межах документів. Інтеграція моделей, зокрема Word2Vec та FastText, дозволить створювати щільні вектори для слів, враховуючи їхнє оточення в контексті, що покращує тематичне групування. Натомість, контекстуальні трансформери, як-от BERT (Bidirectional Encoder

Representations from Transformers) та RoBERTa (Robustly optimized BERT approach), оперують контекстно-залежними уявленнями про слова, здатними відображати більш глибокі лексико-семантичні зв’язки. Кожен із зазначених методів має свої переваги й обмеження – статистичні підходи є обчислювально простішими, але менш точними, у той час як трансформерні моделі демонструють вищу якість, проте вимагають значних обчислювальних ресурсів.

Оцінка тематичної подібності між векторами здійснюється за допомогою різноманітних метрик. Найпоширенішою є косинусна подібність, яка визначає кут між векторами у векторному просторі, незалежно від їхньої довжини. Евклідова та манхеттенська відстані враховують абсолютні різниці між координатами векторів, однак можуть бути менш стабільними при роботі з нормалізованими текстовими даними. Також ефективним є застосування методу k-ближчих сусідів (k-NN), який дозволяє здійснювати класифікацію чи рекомендацію на основі схожості об’єкта до k найближчих зразків у навчальному наборі. Використання цих метрик дозволяє формалізувати процес виявлення тематичних зв’язків між відео та підвищити ефективність автоматизованих систем обробки контенту.

2.4 Вибір стеку технологій для розробки вебсервісу

2.4.1 Технологічні основи розробки вебсервісу

Ефективне створення вебсервісу для інтелектуального пошуку споріднених відео, такого як «YTSimilar», вимагає зваженого вибору технологій, які здатні забезпечити надійну, масштабовану та високопродуктивну роботу системи. Важливими складовими є логіка обробки запитів, зберігання та доступ до даних, а також інфраструктура для хмарного розгортання та подальшого обслуговування. Особливу увагу слід

приділяти вимогам до технологічного стеку, який повинен забезпечувати високу швидкодію в режимі реального часу, сумісність із зовнішніми API (зокрема YouTube Data API), підтримку бібліотек машинного навчання, здатність до масштабування під час роботи з великими обсягами інформації, а також простоту розгортання у хмарних середовищах.

Для досягнення поставлених цілей доцільно використовувати мову програмування Python у поєднанні з фреймворками Flask або FastAPI, системи управління базами даних для збереження інформації про відео, а також інструменти контейнеризації Docker із розгортанням у хмарних сервісах, таких як AWS або Google Cloud. Такий підхід дозволяє побудувати сучасний вебсервіс з надійною архітектурою, здатною адаптуватися до динамічних потреб користувачів.

Вибір відповідного стеку технологій є критично важливим етапом у процесі створення вебсервісу «YTSimilar» для інтелектуального пошуку споріднених відео на платформі YouTube. Успішна реалізація такого сервісу потребує гнучкої, масштабованої та продуктивної архітектури, здатної ефективно обробляти запити користувачів у реальному часі, взаємодіяти з зовнішніми API та забезпечувати стабільну роботу в хмарному середовищі.

Python є однією з найбільш популярних мов програмування, що широко використовується в різних сферах, зокрема в обробці даних, машинному навчанні та розробці вебсервісів. Вибір Python для розробки вебсервісу рекомендацій «YTSimilar» для інтелектуального пошуку споріднених відео на YouTube обґрунтовано низкою факторів, які роблять його оптимальним рішенням для цього проєкту.

Python є стандартом у сфері обробки даних і машинного навчання завдяки своїй простоті, потужним бібліотекам та підтримці численних фреймворків для побудови алгоритмів. Мова надає розробникам доступ до таких інструментів, як NumPy, Pandas, Scikit-learn, TensorFlow та PyTorch, які дозволяють ефективно обробляти великі обсяги даних та реалізовувати складні моделі машинного навчання для персоналізованих рекомендацій.

Для задачі пошуку споріднених відео на YouTube Python забезпечить високу продуктивність і зручність в обробці великих масивів відеоінформації.

Для розробки рекомендаційних систем Python пропонує широкий вибір бібліотек, спеціалізованих на побудові алгоритмів рекомендацій, таких як Surprise, LightFM, Implicit та інші. Ці бібліотеки містять вбудовані алгоритми для колаборативної фільтрації, контентної фільтрації та комбінованих підходів, що дозволяє швидко і ефективно реалізувати рекомендаційні моделі для відео на платформі YouTube. Крім того, Python дозволяє легко налаштовувати та адаптувати ці алгоритми для специфічних потреб вебсервісу, таких як інтеграція з YouTube API, та персоналізації рекомендацій для кожного користувача.

Python має безліч готових бібліотек для інтеграції з різними зовнішніми сервісами, що робить його ідеальним вибором для роботи з YouTube Data API. За допомогою таких бібліотек, як google-api-python-client, можна легко підключитися до YouTube, отримувати дані про відео, канали, коментарі, рейтинги та інші метадані, що дозволяє будувати потужні рекомендаційні системи на основі інформації з платформи. Python забезпечує зручну обробку цих даних і їх подальшу інтеграцію з алгоритмами рекомендацій, що значно спрощує розробку вебсервісу «YTSimilar».

Flask та FastAPI – це два сучасні фреймворки для розробки вебдодатків на мові програмування Python, які активно використовуються для побудови RESTful API. Обидва інструменти мають власні переваги та особливості, що робить їх популярними серед розробників. Flask – це легкий та мінімалістичний фреймворк, який забезпечує розширювану архітектуру і велику кількість додаткових бібліотек для вирішення широкого спектра завдань. Він має довгу історію розвитку, активну спільноту та перевірену стабільність, що робить його хорошим вибором для проєктів різної складності. FastAPI, у свою чергу, є більш новим фреймворком, орієнтованим на високу продуктивність і типізацію. Він базується на

сучасних можливостях Python (асинхронність, анотації типів) і забезпечує автоматичну генерацію документації до API, що спрощує розробку та тестування.

У контексті розробки вебсервісу «YTSimilar» для інтелектуального пошуку споріднених відео на YouTube вибір було зроблено на користь Flask. Основною причиною такого рішення є простота та гнучкість реалізації, яку надає Flask, а також його широка підтримка в контексті інтеграції з моделями машинного навчання. Flask дозволяє швидко розгорнути RESTful API, що необхідне для обробки запитів користувача, взаємодії з YouTube Data API та подання результатів у вигляді рекомендованих відео. Для цілей цієї роботи, де ключовим завданням є обробка запитів до моделі рекомендацій і робота з зовнішніми API, якщо продуктивність FastAPI не є критичним фактором, то стабільність і зрозуміла структура Flask має більш вагомі переваги.

Крім того, Flask добре поєднується з інструментами машинного навчання, такими як Scikit-learn, TensorFlow або PyTorch, дозволяючи без додаткових ускладнень розгортати моделі у вигляді вебсервісу. Простість у налаштуванні маршрутів, обробці запитів та відповіді, а також активна документація робить Flask зручним рішенням для реалізації MVP (мінімально життєздатного продукту) системи рекомендацій. Таким чином, Flask забезпечує баланс між функціональністю, зручністю розробки та достатнім рівнем продуктивності, необхідним для створення ефективного та масштабованого сервісу на кшталт «YTSimilar».

У межах реалізації даного програмного забезпечення система управління базами даних (СУБД) не використовується, оскільки специфіка завдання не передбачає необхідності у довготривалому збереженні, структурованому управлінні або масштабованій обробці великих обсягів інформації. Замість цього для зберігання проміжних результатів аналізу, зокрема транскрибації [16], аудіофайлів або значень схожості, застосовуються базові внутрішні структури мови програмування Python,

такі як словники (dict) та списки (list), що функціонують в оперативній пам’яті під час роботи серверного застосунку. Такий підхід є доцільним для локального проєкту, орієнтованого на обробку обмеженої кількості відео в режимі реального часу. Збереження даних відбувається тимчасово, лише в межах поточної сесії, без запису на диск чи використання зовнішніх сховищ, що дозволяє забезпечити високу швидкодію та мінімізувати складність архітектури. У разі перезапуску сервера всі збережені об’єкти очищаються, що унеможливлює накопичення даних, але водночас усуває потребу в контролі цілісності або узгодженості інформації.

З точки зору архітектури серверної частини, система будується на модульному підході. Центральним компонентом є веб-сервер, реалізований на Flask, який обробляє вхідні запити від користувача через RESTful API. Запит спочатку надходить до API-обробника, який надсилає запит до Flask-серверу з ID відео. Далі, дані передаються до модуля з вбудованою ML-моделлю, яка здійснює пошук схожих відео на основі векторних представлень. Алгоритм роботи такої архітектури зображено на рисунку 2.9.

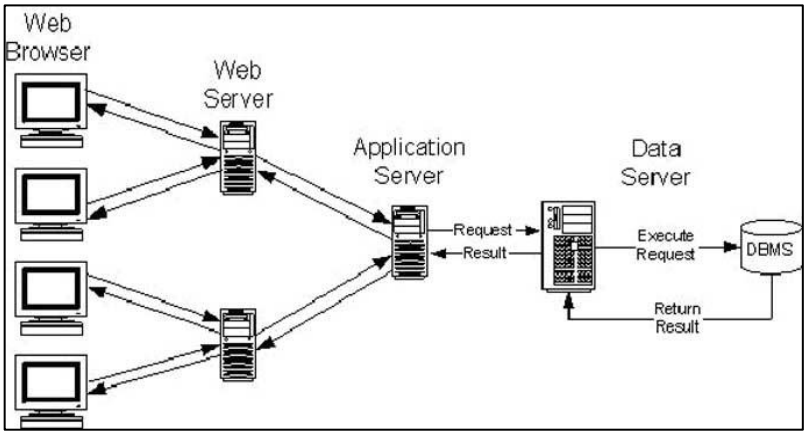

Рисунок 2.9 – Багаторівнева архітектура або N-рівнева архітектура

Таким чином, обрана база даних у поєднанні з модульною архітектурою дозволяє реалізувати масштабований, швидкий та гнучкий вебсервіс, оптимізований під задачі рекомендаційного характеру.

2.4.2 Хмарні технології та розгортання

У процесі створення вебсервісу «YTSimilar» для інтелектуального пошуку споріднених відео на YouTube виникає необхідність у використанні хмарних технологій. Основними причинами такого вибору є потреба в масштабованості, гнучкості інфраструктури та готовності системи до обробки змінного навантаження. Хмарні сервіси дозволяють ефективно розподіляти обчислювальні ресурси, автоматично масштабуючи інфраструктуру у разі збільшення кількості користувачів чи запитів, що є критично важливим для стабільної роботи рекомендаційної системи в режимі реального часу. Крім того, завдяки можливостям безперервної інтеграції (CI) та автоматичного розгортання (CD), хмарні платформи сприяють швидшій доставці нових версій програмного забезпечення, спрощують тестування, оновлення і моніторинг роботи сервісу.

У цьому контексті доцільним є використання контейнеризації за допомогою Docker, що дозволяє створювати ізольовані середовища для кожного з компонентів вебсервісу. Такий підхід забезпечує однаковість роботи застосунку в різних середовищах – на локальній машині розробника, у тестовому середовищі та на продуктивному сервері. Docker спрощує процес розгортання, дозволяє швидко відтворювати конфігурації, масштабувати компоненти системи та забезпечує кросплатформенність, що є важливою перевагою в умовах динамічної розробки.

При порівнянні провідних хмарних платформ – AWS (Amazon Web Services) та Google Cloud Platform (GCP) ключовим фактором на користь GCP. З огляду на належність обох платформ корпорації Google, забезпечує стабільну та високошвидкісну взаємодію з YouTube Data API – основним джерелом інформації для аналізу та пошуку відео. GCP підтримує зручне розгортання Python/Flask-додатків через App Engine або Cloud Run, що спрощує майбутнє масштабування без складної конфігурації.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмної системи

Розроблене програмне забезпечення є веброзширенням для пошуку схожих відео на платформі YouTube на основі їх транскриптів. Система реалізована за принципом клієнт-серверної архітектури, де клієнтським компонентом виступає розширення для браузеру на рушії Chromium такого як Google Chrome чи Opera. Серверною частиною виступає вебсервер написаний на Flask.

Клієнтська частина відповідає за взаємодію з користувачем, ідентифікацію відео з активної вкладинки браузера , надсилання запиту до серверного API та відображення результатів роботи.

Серверна частина виконує обробку запиту, отримує транскрипти відео, проводить сематичне порівняння з іншими відео та формує відповідь у форматі JSON об’єкта.

3.2 Клієнтська частина

Клієнтський інтерфейс (рисунок 3.1).реалізовано як розширення для браузеру, що складається з HTML-файлу (popup.html) та скрипта (popup.js).

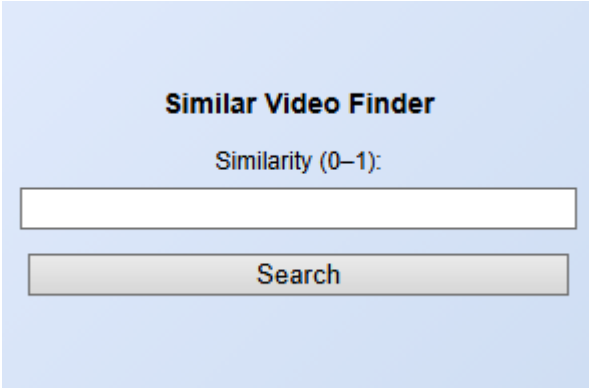

Рисунок 3.1 – Інтерфейс розширення

Користувачеві доступний простий інтерфейс, у якому можна встановити поріг схожості відеоматеріалів та натиснути кнопку пошуку

Перед тим як дати результат в інтерфейсі розширення, скрипт виконує такі дії:

– зчитує поточне посилання у вкладці;

– отримує посилання на відео з поточної сторінки YouTube;

– надсилає запит до локального сервера Flask;

– виводить результати, відео з найбільшим рівнем подібності (рисунок 3.2).

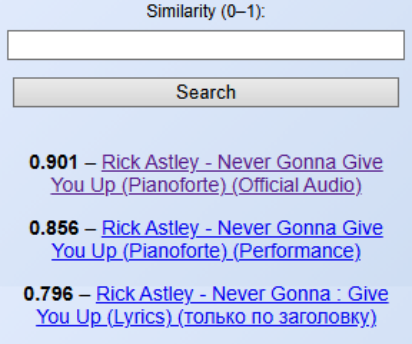

Рисунок 3.2 – Інтерфейс після успішного пошуку

Приклад та вид розширення на сторінці відео Youtube (рисунок 3.3).

Рисунок 3.3 – Вид розширення на сторінці відео

3.3 Серверна частина

Серверна частина реалізовано на мові програмування Python з використанням фреймворку Flask. Основний функціонал знаходиться у файлі app.py, де реалізована більша частина коду. До ключових елементів реалізації можна виділити отримання транскриптів, семантичне порівняння, вибір кандидатів та Rest API.

3.3.1 Отримання транскриптів

Для отримання субтитрів відео використовується бібліотека youtube-transcript-api. Якщо автоматизований запит не дає жодних результатів, то виконується спроба завантажити .vtt файли за допомогою yt-dlp із використанням cookies-файлу.

3.3.2 Семантичне порівняння

Схожість відео визначається за допомогою обчислення косинусної подібності між векторними представленнями транскриптів. Для цього зі складу бібліотеки sklearn використовується TfidfVectorizer та за допомогою коефіцієнта Левенштейна (Levenshtein.ratio) порівнюються заголовки для більш точного порівняння.

3.3.3 Вибір кандидатів

З метою скорочення пошукового простору, з оригінального транскрипту випадковим чином вибираються декілька фраз певної довжини. За кожною фразою виконується пошук відео на YouTube за допомогою Google API та після цього транскрипти кандидатів порівнюються з оригіналом відео.

3.4 Використані технології та бібліотеки

У процесі реалізації були використані такі основні засоби:

– Flask – для створення серверного RestAPI;

– youtube-transcript-api – для отримання транскриптів;

– yt-dlp – для завантаження vtt-файлів як резервним механізм;

– Google API – для пошуку відео на YouTube;

– Levenshtein – для порівняння назв відео;

– Scikit-learn – для обчислення схожості транскриптів.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено та впроваджено інноваційний вебсервіс рекомендацій YTSimilar, призначений для інтелектуального пошуку тематично споріднених відео на платформі YouTube. Ключова відмінність даного рішення полягає у фокусі на глибинний аналіз відеоконтенту шляхом транскрибації, аналізу та обробки транскриптів, тегів та метаданих. Це забезпечує значно вищу точність рекомендацій у порівнянні з традиційними алгоритмами, які переважно ґрунтуються на «поверхневих» (явних) ознаках відео.

Одним із головних досягнень роботи є обґрунтування архітектурного підходу до реалізації сервісу. Було створено клієнт-серверну систему, де роль клієнта виконує браузерне розширення, а серверна частина реалізована у вигляді RESTful API на основі фреймворку Flask. Для забезпечення доступу до необхідних даних було інтегровано YouTube Data API (для збору метаданих) та Google Speech-to-Text (для автоматичної генерації транскрипцій). Із метою масштабування та спрощення розгортання, застосовано технологію контейнеризації Docker і використано можливості хмарної платформи Google Cloud Platform.

Алгоритмічна основа системи базується на гібридному підході до формування рекомендацій. Було поєднано методи семантичного аналізу на основі TF-IDF та обчислення косинусної подібності з контекстним порівнянням заголовків за допомогою коефіцієнта Левенштейна. Для підвищення ефективності пошуку реалізовано механізм вибірки кандидатів – здійснюється випадковий вибір ключових фраз із транскрипту з подальшим виконанням пошуку через YouTube API. У разі недоступності субтитрів реалізовано відмовостійкий механізм перемикання між джерелами транскрипції (youtube-transcript-api та yt-dlp), що підвищує надійність функціонування сервісу.

Значна увага приділялася покращенню користувацького досвіду. Інтерфейс браузерного розширення розроблено з урахуванням принципів інтуїтивності та зручності, зокрема користувачеві надано можливість регулювання порогу схожості відео. Завдяки кешуванню проміжних результатів та оптимізації обчислювального пайплайну машинного навчання забезпечено обробку запитів у режимі реального часу.

Науково-практична значимість розробки полягає у доведенні ефективності підходу, орієнтованого на транскрипти, як джерела глибокого семантичного аналізу відеоконтенту. Проведене порівняння показало, що точність рекомендацій, сформованих на основі транскриптів, зросла у порівнянні з методами, які базуються виключно на метаданих (заголовках, тегах). Крім того, запропонований сервіс дозволив частково подолати явище «інформаційної бульбашки», характерне для стандартних алгоритмів фільтрації YouTube, шляхом виявлення тематично близького, проте менш популярного, або забороненого в регіоні контенту.

Серед перспектив подальшого розвитку сервісу слід виділити впровадження глибоких моделей семантичного аналізу, таких як BERT чи SentenceTransformers, що дозволить підвищити точність векторного подання змісту. Також передбачається реалізація персоналізованих рекомендацій на основі історії переглядів користувача, розширення функціональності шляхом підтримки інших платформ (наприклад, TikTok, Vimeo) а також оптимізація архітектури через перехід на FastAPI і використання Redis для кешування векторних подань.

Таким чином, у ході роботи було повністю виконано поставлені завдання, а розроблений вебсервіс YTSimilar можна вважати успішним прикладом інтеграції сучасних технологій машинного навчання у практичні рішення, орієнтовані на кінцевого користувача. Результати дослідження засвідчують як прикладну цінність сервісу для освітньої та маркетингової сфери, так і його науковий потенціал у межах досліджень у галузях обробки природної мови, систем рекомендацій і хмарних обчислень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. An overview on application of machine learning techniques in optical networks / F. Musumeci et al. *arXiv*. 2018. Vol. 1, no. 2. P. 2–5.

2. Improving Training Stability for Multitask Ranking Models in Recommender Systems / J. Tang et al. *KDD '23: The 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Long Beach CA USA. New York, NY, USA, 2023. URL: https://doi.org/10.1145/3580305.3599846 (date of access: 10.05.2025).

3. Le Merrer E., Tredan G., Yesilkanat A. Modeling rabbit-holes on YouTube. *Social Network Analysis and Mining*. 2023. Vol. 13, no. 1. URL: https://doi.org/10.1007/s13278-023-01105-9 (date of access: 10.05.2025).

4. A Utility-Preserving Obfuscation Approach for YouTube Recommendations / J. Zhang et al. *Proceedings on Privacy Enhancing Technologies*. 2023. Vol. 2023, no. 4. P. 522–539. URL: https://doi.org/10.56553/popets-2023-0123 (date of access: 10.05.2025).

5. Khadka P., Lamichhane P. Content-based Recommendation Engine for Video Streaming Platform. *arXiv*. 2023. URL: https://arxiv.org/abs/2308.08406.

6. Contributors to Wikimedia projects. Recommender system - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Recommender_system (date of access: 10.05.2025).

7. Contributors to Wikimedia projects. ACM Conference on Recommender Systems - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/ACM_Conference_on_Recommender_Systems (date of access: 10.05.2025).

8. Contributors to Wikimedia projects. Collaborative filtering - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Collaborative_filtering (date of access: 10.05.2025).

9. Inc. G. Deep Neural Networks for YouTube Recommendations. Google Inc., 2016. URL: https://research.google.com/pubs/archive/45530.pdf.

10. Hafeez Y. The Science Behind YouTube Algorithm in 2025 [Updated]. *Social Champ*. URL: https://www.socialchamp.com/blog/youtube-algorithm/ (date of access: 10.05.2025).

11. Evaluating Recommendation Systems - Precision@k, Recall@k, and R-Precision | Shaped Blog. *Shaped | Recommendations and Search*. URL: https://www.shaped.ai/blog/evaluating-recommendation-systems-part-1 (date of access: 10.05.2025).

12. Colour spaces - a review of historic and modern colour models* | Hastings | African Vision and Eye Health. *Home Page*. URL: https://doi.org/10.4102/aveh.v71i3.76 (date of access: 10.05.2025).

13. Sherstinsky A. Redirecting. *Home Page*. URL: https://doi.org/10.1016/j.physd.2019.132306 (date of access: 10.05.2025).

14. Stevens J. Why you need metadata for Big Data success - DataScienceCentral.com. *Data Science Central*. URL: https://www.datasciencecentral.com/profiles/blogs/why-you-need-metadata-for-big-data-success (date of access: 10.05.2025).

15. Bonchek M. Little Data Makes Big Data More Powerful. *Harvard Business Review*. URL: https://hbr.org/2013/05/little-data-makes-big-data-mor (date of access: 10.05.2025).

16. Салтан С. О., Гриньова О. Є. Nlp- та LLM-орієнтована система персоналізованих рекомендацій фільмів. *Радіоелектроніка та молодь у XXI столітті : матеріали 29-го Міжнар. молодіж. форуму, 16 – 19 квітня 2025 р.* Харків, 2025. Т. 6. С. 56–58.