

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО
СИСТЕМНОГО АНАЛІЗУ**

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

« ____ » _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему:

**«Розробка рекомендаційної системи з використанням методів
машинного навчання та статистичного аналізу»**

Виконав:

студент 4 курсу, групи КА-04

Бородатюк Дмитро Михайлович _____

Керівник:

асистент, Соболь Ольга Олександрівна _____

Консультант з економічного розділу:

д.ф.е., доцент Мажара Гліб Анатолійович _____

Консультант з нормоконтролю:

к.ф.-м.н. Статкевич Віталій Михайлович _____

Рецензент:

доцент кафедри ІШ, д-р філософії, Гуськова Віра Геннадіївна _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» _____ 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Бородатюку Дмитру Михайловичу

1. Тема роботи «Розробка рекомендаційної системи з використанням методів машинного навчання та статистичного аналізу», керівник роботи асистент Соболь Ольга Олександрівна, затверджені наказом по університету від «31» травня 2024 р. № _____
2. Термін подання студентом роботи 11.06.2024.
3. Вихідні дані до роботи: набори даних, які містять відомості про інформацію популярних кінофільмів.
4. Зміст роботи: дослідження предметної області, аналіз засобів реалізації проекту, розробка архітектури рекомендаційної системи з використанням методів машинного навчання та статистичного аналізу на прикладі відеострімінгового сервісу.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): рисунки демонстрації роботи продукту, таблиці порівнянь результатів різних методів, схема «Основні типи рекомендаційних систем», схема «Блок-схема реалізації продукту», презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	д.ф.е., доцент Мажара Гліб Анатолійович		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури за темою	10.04.2024	Виконано
2.	Підготовка першого розділу	18.04.2024	Виконано
3.	Підготовка другого розділу	23.04.2024	Виконано
4.	Розробка програмного продукту	03.05.2024	Виконано
5.	Підготовка третього розділу	14.05.2024	Виконано
6.	Підготовка четвертого розділу	22.05.2024	Виконано
7.	Оформлення розділів відповідно до нормоконтролю	25.05.2024	Виконано
8.	Оформлення дипломної роботи	27.05.2024	Виконано
9.	Підготовка презентації доповіді	29.05.2024	Виконано

Студент

Дмитро БОРОДАТЮК

Керівник

Ольга СОБОЛЬ

РЕФЕРАТ

Дипломна робота містить: 127 с., 12 табл., 50 рис., 2 дод., 22 джерела.

РЕКОМЕНДАЦІЙНІ СИСТЕМИ, КОНТЕНТНА ФІЛЬТРАЦІЯ,
КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ОБРОБКА ДАНИХ, ОБРОБКА
МЕТАІНФОРМАЦІЇ.

Мета роботи – порівняння різних методів та підходів до створення рекомендаційних систем, дослідження методів машинного навчання та статистичного аналізу, особливостей їх роботи та розробка програмного продукту на прикладі рекомендаційної системи відеострімінгового сервісу.

Об'єкт дослідження – процес формування рекомендацій на прикладі кінофільмів.

Предмет дослідження – методи машинного навчання та статистичного аналізу і використання їх в задачах генерації рекомендацій.

У роботі проведено аналіз та порівняння типів та методів генерацій рекомендацій. В ході виконання дипломної роботи були детально розібрані методи машинного навчання та статистичного аналізу, було проаналізовано їх особливості, принцип роботи, переваги та недоліки.

При виконанні подальшого розвитку предмету дослідження доцільно розробити моделі, які будуть сумісні з автоматичним оновленням бази даних та дозволять максимізувати точність рекомендації.

ABSTRACT

The thesis contains: 127 pages, 12 tables, 50 figures, 2 appendices, 22 references.

RECOMMENDATION SYSTEMS, CONTENT-BASED FILTERING, COLLABORATIVE FILTERING, DATA PROCESSING, METADATA PROCESSING.

The purpose of the research is to compare different methods and approaches to creating recommendation systems, to study machine learning and statistical analysis methods, their operational features, and to develop software based on the example of a video streaming service recommendation system.

The object of the research is the process of forming recommendations using the example of movies recommender.

The subject of the research is the set of machine learning and statistical analysis method and their use in recommendation generation tasks.

The work analyzes and compares types and methods of recommendation generation. In the course of the thesis, machine learning and statistical analysis methods were thoroughly examined, their features, operating principles, advantages, and disadvantages were analyzed.

In further research, it is advisable to develop models that are compatible with automatic database updates and allow for maximizing the accuracy of recommendations.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Загальний аналіз предметної області	10
1.2 Актуальність рекомендаційних систем.....	11
1.3 Типи рекомендаційних систем.....	14
1.3.1 Системи на основі вмісту	15
1.3.2 Колаборативне фільтрування.....	17
1.3.3 Гібридні системи.....	21
1.3.4 Системи на основі знань.....	23
1.4 Постановка задачі дослідження	24
1.5 Висновки до розділу 1	25
РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ.....	26
2.1 Аналіз рекомендаційних систем відомих сервісів.....	26
2.1.1 Аналіз рекомендаційних систем Airbnb	27
2.1.2 Аналіз рекомендаційних систем LinkedIn.....	28
2.1.3 Аналіз рекомендаційних систем Netflix	30
2.1.4 Аналіз рекомендаційних систем MovieLens	32
2.1.5 Висновок порівняльного аналізу систем	33
2.2 Аналіз критеріїв якості для рекомендаційних систем.....	34
2.3 Методи векторизації та їх аналіз	39
2.4 Методи машинного навчання	44
2.4.1 Кластеризація K-Means	44
2.4.2 Метод головних компонентів K-Means	47
2.4.3 Метод самоорганізуючих карт з аналізом головних компонентів ..	49
2.4.4 Метод k-найближчих сусідів	50
2.5 Висновки до розділу 2	53
РОЗДІЛ 3 РОЗРОБКА АРХІТЕКТУРИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ НА ПРИКЛАДІ СЕРВІСУ ВІДЕОСТРІМІНГУ	54

3.1 Вступ.....	54
3.2 Розробка архітектури рекомендаційної системи на основі content-based моделі	54
3.3 Огляд використаних бібліотек	57
3.4 Опис вхідних даних.....	60
3.5 Етапи створення реалізованого продукту.....	62
3.5.1 Обробка даних	62
3.5.2 Обробка метаінформації.....	64
3.5.3 Векторизація тексту	68
3.5.4 Функції рекомендації.....	70
3.6 Аналіз результатів роботи реалізованого продукту	72
3.7 Розробка інтерфейсу та приклад візуалізації	87
3.8 Висновки до розділу 3	90
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	91
4.1 Формування завдання проектування.....	91
4.2 Обґрунтування функцій програмного продукту.....	92
4.3 Обґрунтування системи параметрів програмного продукту	95
4.4 Аналіз експертного оцінювання параметрів	98
4.5 Аналіз рівня якості варіантів реалізації функцій.....	101
4.6 Економічний аналіз варіантів розробки ПП.....	102
4.7 Вибір кращого варіанта техніко-економічного рівня	106
4.8 Висновки до розділу 4	107
ВИСНОВКИ	108
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	110
ДОДАТОК А.....	113
ДОДАТОК Б.....	118

ВСТУП

Рекомендаційні системи відіграють важливу роль у сприянні зростанню електронної комерції в багатьох веб-додатках [17]. Також, рекомендаційна система є однією з найважливіших дослідницьких областей сьогодення, оскільки вона допомагає користувачам знаходити свої інтереси в Інтернеті.

Багато продуктів купуються онлайн, і є зростаючий попит клієнтів на фільтрацію великої кількості товарів, доступних на веб-сайтах, щоб конкретний товар можна було легше знайти відповідно до їхніх фактичних інтересів.

Рекомендаційні системи використовуються для прогнозування оцінки, яку користувач дасть товару, якщо він бачить його вперше на сайті, використовуючи системи фільтрації інформації з історії оцінок користувача на сайті або знаходячи специфікації схожих товарів, або знаходячи спільні інтереси, використовуючи демографічну інформацію з їхнього профілю.

Рекомендаційні системи мають різні типи фільтрації, такі як (колабораційна, на основі вмісту та гібридна) для створення ефективного рекомендаційного механізму. Якщо користувач купує товар на сайті, йому будуть рекомендовані інші товари на основі специфікацій товару, який він купив. Наприклад, якщо він купив книгу, система порекомендує йому книги на ту саму тему або більше книг, написаних тими самими авторами.

Використання рекомендаційних систем несе свої переваги для бізнесу. Серед них це підвищення дохідності — в останні роки багато дослідників вивчали та створювали багато алгоритмів для збільшення кількості онлайн-клієнтів, як, наприклад, на сайті Amazon відбувся зріст на 35% [11]. Зростання задоволеності клієнтів — часто клієнти очікують побачити рекомендації схожих продуктів на основі їхнього останнього пошуку на сайті. Головним чином тому, що вони вірять, що отримають більше шансів знайти кращі

продукти. Збережені дані переглядів з попереднього сеансу або список переглянутих товарів користувачів, коли вони залишають сайт та повертаються пізніше, допомагають електронній комерції утримувати клієнтів та підвищувати їх залученість.

Метою даної роботи є створення рекомендаційної системи на основі методів машинного навчання та статистичного аналізу для генерації рекомендацій фільмів. Система отримуватиме метадані фільму, на основі якого необхідно створити рекомендацію, та формуватиме вихідні результати за допомогою спеціального алгоритму у вигляді переліку подібних фільмів.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний аналіз предметної області

Рекомендаційні системи – це інформаційні системи, які пропонують користувачам релевантні продукти, послуги чи контент на основі їхніх інтересів, поведінки та уподобань. Ці системи використовуються в широкому спектрі галузей, включаючи електронну комерцію, потокове мультимедіа, музичні сервіси, новинні агрегатори та соціальні мережі.

Основна мета рекомендаційних систем – персоналізувати досвід користувачів, пропонуючи їм те, що їм, ймовірно, сподобається або буде корисно. Це може призвести до збільшення залученості, покращення конверсії та зростання доходу для компаній, які їх використовують.

Рекомендаційні системи ґрунтуються на різних алгоритмах, які аналізують дані про користувачів, такі як історія покупок, переглянуті сторінки, лайки та коментарі. Ці алгоритми можуть використовувати різні методи, такі як фільтрування на основі вмісту, співвідношення користувачів та демографічне фільтрування.

Деякі з ключових компонентів рекомендаційних систем включають:

- збір даних: збирання даних про користувачів та їхню поведінку;
- підготовка даних: очищення та обробка даних для їх використання в алгоритмах рекомендацій;
- алгоритми рекомендацій: алгоритми, які аналізують дані та генерують рекомендації;
- оцінка та вдосконалення: оцінка ефективності системи та внесення необхідних змін для її покращення.

Розглядаючи сутність рекомендаційних систем, важливо відзначити їхню глибоку інтеграцію в сучасне цифрове середовище. Ці системи не лише є інструментом надання рекомендацій, але й важливою складовою частиною

стратегії бізнесу в онлайн-сфері. Вони відіграють роль не лише в покращенні користувацького досвіду, але й в оптимізації процесів продажу, маркетингу та взаємодії з клієнтами.

Одним з ключових аспектів роботи рекомендаційних систем є їх здатність до аналізу та обробки великих обсягів даних. Вони опрацьовують інформацію про користувачів, їх взаємодію з платформою чи сервісом, їх вподобання та інші параметри, щоб забезпечити найбільш релевантні рекомендації.

Крім того, рекомендаційні системи дозволяють підприємствам створювати персоналізовані стратегії взаємодії з кожним користувачем. Завдяки аналізу даних, ці системи можуть пропонувати індивідуально настроєні пропозиції, що підвищує ефективність комунікації та залучення клієнтів. Більш того, рекомендаційні системи допомагають покращити рівень обслуговування користувачів, надаючи їм доступ до більш релевантної та цікавої інформації. Це сприяє підвищенню задоволення від використання платформи чи сервісу та збільшенню лояльності клієнтів.

Отже, рекомендаційні системи стали не просто інструментом в області інформаційних технологій, але й невід'ємною складовою частиною стратегії бізнесу в цифровому світі. Вони допомагають підприємствам досягати більшої ефективності та конкурентоспроможності, а також забезпечують користувачам персоналізований та задовільний досвід взаємодії з продуктом чи послугою.

1.2 Актуальність рекомендаційних систем

Сьогодні, у світі величезних обсягів інформації та широкого спектру доступних товарів і послуг, рекомендаційні системи стали невід'ємною частиною багатьох онлайн-сервісів. Рекомендаційні системи використовуються у багатьох сферах, таких як електронна комерція, медіа-сервіси, соціальні мережі та стрімінгові платформи. Вони допомагають

користувачам знаходити нові товари, послуги чи контент, які відповідають їхнім інтересам та потребам.

Однією з ключових причин актуальності рекомендаційних систем є зростання обсягів даних. В умовах, коли кількість доступної інформації швидко збільшується, стає дедалі важче знайти потрібну інформацію без допомоги автоматизованих систем. Рекомендаційні системи, використовуючи різні методи машинного навчання та аналізу даних, дозволяють ефективно вирішувати цю проблему.

Рекомендаційні системи активно використовуються провідними компаніями по всьому світу [17]. Оглянемо найвідоміші приклади.

1. Amazon: використовує рекомендаційні системи для персоналізації покупок, тому їхні алгоритми аналізують історію покупок користувачів і рекомендують товари, які можуть їх зацікавити. Алгоритм цієї моделі зображено на рис. 1.1:

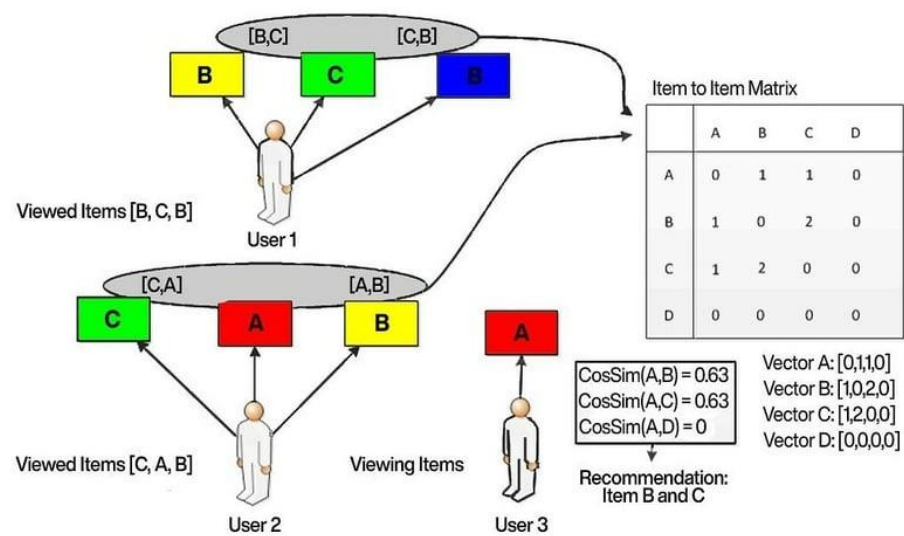


Рисунок 1.1 – Алгоритм рекомендаційної системи Amazon [17]

2. Netflix: застосовує складні моделі для рекомендації фільмів і серіалів на основі переглядів користувачів, їх оцінок і поведінкових даних.

3. Spotify: використовує рекомендаційні системи для створення персоналізованих плейлистів та рекомендацій нових треків, як зображено на рис. 1.2 та на основі музичних вподобань користувачів.

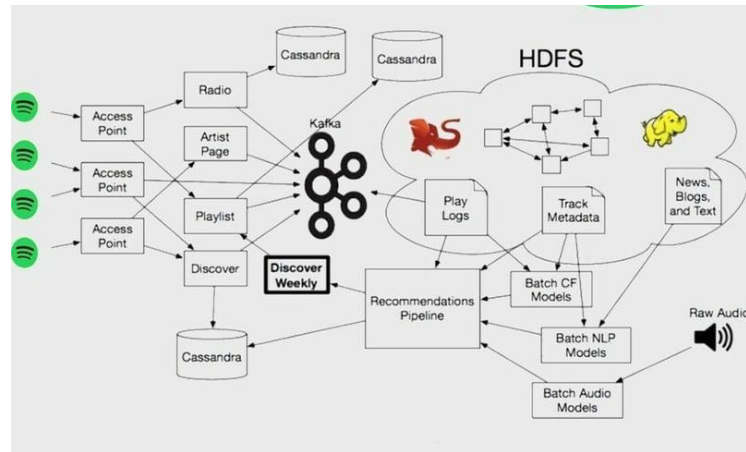


Рисунок 1.2 – Алгоритм створення персоналізованих рекомендацій Spotify [17]

4. YouTube: алгоритми рекомендують відео на основі переглядів, пошукових запитів та інших взаємодій користувачів з платформою використовуючи різні параметри, як зображено на рис. 1.3:

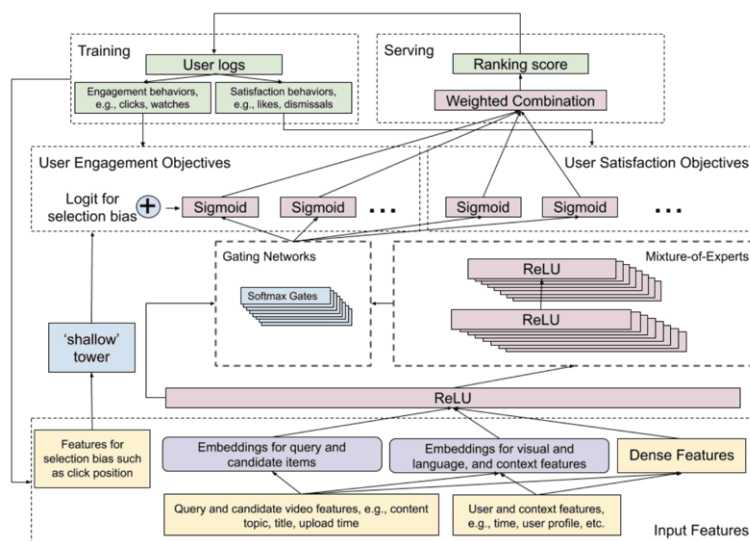


Рисунок 1.3 – Використання параметрів для створення рекомендацій YouTube [17]

5. Facebook: використовує рекомендаційні системи для персоналізації стрічки новин, реклами та пропонуванних друзів.

Рекомендаційні системи значно покращують користувацький досвід, роблячи його більш індивідуальним і зручним. Вони сприяють збільшенню залученості користувачів і підвищенню рівня задоволеності від використання сервісів. Для бізнесу це означає підвищення рівня конверсії та утримання клієнтів, а також зростання прибутків.

Попри значні успіхи у розвитку рекомендаційних систем, існують і певні виклики. Однією з головних проблем є питання конфіденційності та безпеки даних користувачів. Іншою важливою проблемою є ефект "інформаційної бульбашки", коли користувачі отримують лише ті рекомендації, які відповідають їхнім попереднім вподобанням, що обмежує їхній кругозір.

Водночас, з розвитком технологій машинного навчання та великих даних, рекомендаційні системи мають величезний потенціал для подальшого вдосконалення. Впровадження нових методів, таких як глибоке навчання, може значно підвищити точність і ефективність рекомендацій, роблячи їх ще більш корисними для користувачів і бізнесу.

Таким чином, актуальність рекомендаційних систем у сучасному світі важко переоцінити. Вони стали ключовим інструментом для навігації у світі великих даних, забезпечуючи користувачам доступ до найбільш релевантної та цікавої інформації, а компаніям — можливість ефективно взаємодіяти зі своїми клієнтами.

1.3 Типи рекомендаційних систем

Варто розглянути типи даних, на основі яких можна приймати рішення. Це можуть бути дані розділені на групи, що стосуються користувачів та товарів, і в цих категоріях може міститися наступна інформація.

- Інформація про користувачів (user-item interactions): інформація про взаємодію користувачів з продуктами сервісів. Може містити інформацію про рейтинги, оцінки, кількість покупок, персональні вподобання тощо.

- Інформація про характеристики (characteristic information): зазвичай міститься інформація про товари або послуги (в залежності від предмету аналізу) на кшталт категорій, жанрів, ключових слів тощо.

Рекомендаційні системи можна розподіляти на різні категорії в залежності від використання цих даних:

- системи фільтрації на основі вмісту (content-based filtering): використовується інформація про характеристики;

- колаборативне фільтрування (collaborative filtering): використовується інформація про користувачів.

Ці два види систем є стандартними та найпоширенішими, проте також існують і інші види:

- системи на основі знань (knowledge based): використовуються знання про користувача, товар та критерій рекомендації;

- гібридні системи (hybrid systems): використовуються різні комбінації видів інформації з вищенаведених.

Кожен з цих видів систем розглянемо далі.

1.3.1 Системи на основі вмісту

На відміну від систем, що базуються на рейтингах користувачів, системи рекомендацій на основі вмісту не використовують історію оцінок інших користувачів. Замість цього вони спираються на опис елементів контенту, представлений у вигляді ключових слів, тегів, категорій або інших метаданих. Аналізуючи ці дані, система виявляє зв'язки та подібності між елементами, формуючи на їх основі рекомендації для конкретного користувача, як зображено на рис. 1.4.

Системи рекомендацій на основі вмісту є цінним інструментом для персоналізації досвіду користувачів в інформаційних системах. Їхня здатність аналізувати опис контенту та виявляти зв'язки між елементами робить їх ефективними для формування релевантних рекомендацій, особливо для нових користувачів або в холодних стартових сценаріях.

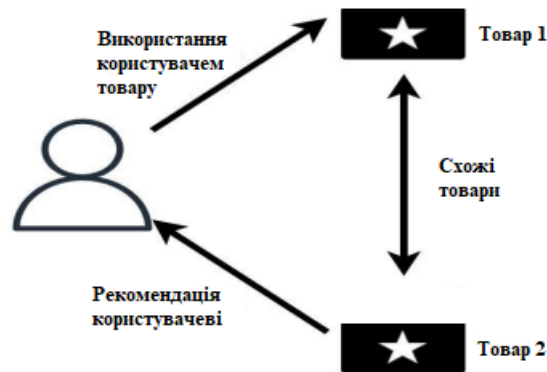


Рисунок 1.4 – схематичне зображення принципу роботи рекомендаційних систем на основі вмісту [6]

Розглянемо систему на основі вмісту з практичної точки зору, як систему з рекомендації фільмів. Спочатку треба виокремити теги для кожного фільму, або ще їх можна назвати атрибутами. Атрибути елементів називаються "вмістом". На основі цих атрибутів класифікують, чи схожі дві речі чи ні. Атрибутами можуть бути слова, зазначені в базі даних, такі як жанр, імена акторів, імена режисерів, опис або будь-які інші метадані. Якщо атрибути збігаються або мають високу схожість, то два фільми можна класифікувати як схожі.

Наприклад, для рекомендації фільмів може використовуватись алгоритм косинусна схожість. Косинусна схожість – це міра, що використовується для визначення схожості між двома елементами. З математичної точки зору її можна визначити як косинус кута між двома векторами у тривимірній площині (1.1):

(1.1)

$$\text{sim}(A, B) = \cos(\theta) = \frac{A * B}{||A|| ||B||}$$

Значення міри може коливатися від -1 до 1, елементи розташовуються у порядку спадання.

Також можемо перевірити евклідову відстань між двома векторами (1.2), щоб визначити, наскільки вони різняться або схожі один на одного:

(1.2)

$$\text{Euclidian Distance} = \sqrt{(x_1 - y_1)^2 + \dots + (x_N - y_N)^2}$$

У нашому випадку одним із векторів є фільм, який шукають, а решту фільмів у базі даних перевіряють як другий вектор. Кількість фільмів, які мають найменшу евклідову відстань від шуканого фільму, відображаються як рекомендації.

Косинусна схожість - це тип фільтрації за вмістом. Це один із найпопулярніших методів, що використовуються в системах рекомендацій. Сенс таких систем рекомендації полягає в тому, що якщо користувачеві сподобався певний товар, йому може сподобатися й товар, схожий на нього.

1.3.2 Колаборативне фільтрування

Рекомендаційні системи на основі колаборативного фільтрування – системи, які будують свої рекомендації на основі вподобань інших користувачів. Ключовою концепцією цих систем є те, що для формування рекомендації приймається до уваги поведінка інших користувачів.

До прикладу, якщо потрібно створити рекомендацію фільму певному користувачеві, то ці системи будуть звертати увагу на реакції на фільми інших

користувачів з подібними вподобаннями, і на основі цього складати рекомендації. Або, якщо потрібно визначити чи сподобається цільовому користувачеві певний фільм, то ці системи будуть оцінювати реакцію інших користувачів на цей фільм. Тобто, методи колаборативного фільтрування прогнозують рейтинг цільового користувача для певного елемента, використовуючи рейтинги схожих користувачів для цього елемента.

Існують два способи застосування колаборативного фільтрування.

1. Методи на основі пам'яті (neighborhood-based) – це простий спосіб використання рейтингів інших користувачів для прогнозування рейтингів. Вони бувають двох видів: на основі користувачів та на основі елементів: колаборативне фільтрування на основі користувачів формує пул схожих користувачів та на основі вподобань одного користувача формується вподобання для всього пулу, тоді як колаборативне фільтрування на основі елементів формує пули схожих елементів на основі оцінок і використовує їх для прогнозування взаємодії з подібними елементами.

2. Методи на основі моделі роблять крок вперед у колаборативному фільтруванні та використовують машинне навчання та ймовірнісні моделі. Деякі їх приклади, включають дерева рішень, моделі з латентними факторами та нейронні мережі як класифікатори типу "чорний ящик".

В алгоритмах на основі користувачів [6], прогнозування елемента (і) для користувача (u) обчислюється шляхом обчислення зваженої суми оцінок елемента, які надали користувачі. Прогноз можна визначити за формулою (1.3):

(1.3)

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}},$$

де $P_{u,i}$ – прогноз для елемента; $r_{v,i}$ – оцінка, яку користувач v дав елементу i.

В алгоритмах на основі елементів знаходиться схожість між кожним набором елементів i , на цій основі, рекомендуються схожі елементи, які користувачі вподобали раніше. Цей алгоритм працює подібно до колаборативного фільтрування за користувачами з однією відмінністю – замість суми рейтингів "сусідів-користувачів" береться сума рейтингів "сусідів-елементів". Прогнозування визначається за формулою (1.4):

(1.4)

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,N}|)}$$

А схожість між елементами (1.5) визначається наступним чином:

(1.5)

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} * \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

Розглянемо колаборативне фільтрування на практичному прикладі [-] рекомендацій фільмів. Припустимо, маємо користувача (K1), який вповодобав 3 фільми (Ф1, Ф2, Ф3). Зафіксуємо це у вигляді $K1 = \Phi1, \Phi2, \Phi3$. Також є другий користувач (K2), який вповодобав фільми 3 (Ф1, Ф3, Ф4), тобто $K2 = \Phi1, \Phi3, \Phi4$. І є третій користувач (K3), який вповодобав лише 1 фільм (Ф1), і маємо надати йому рекомендацію щодо інших фільмів, які йому мають сподобатися, тобто $K3 = \Phi1, \dots?$

В попередньому абзаці міститься інформація, яку використаємо як данні. З цих даних необхідно створити матрицю А, яка буде містити вповодобання користувачів K_i для фільмів Φ_i . Візуалізуємо це на рис. 1.5.

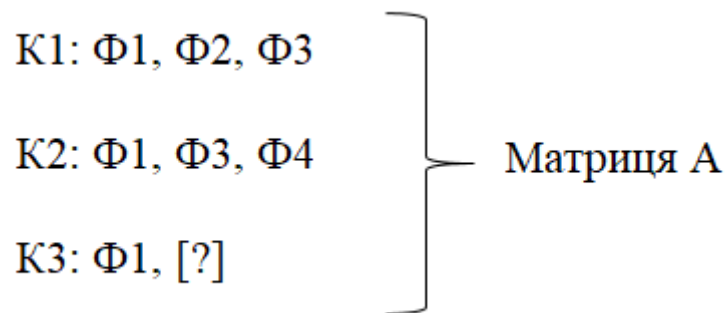


Рисунок 1.5 – схематичне зображення принципу роботи рекомендаційних систем з колаборативним фільтруванням на 1 етапі [4]

Тож, бачимо в даних $K1 = \Phi1, \Phi2, \Phi3$, $K2 = \Phi1, \Phi3, \Phi4$, $K3 = \Phi1, \dots?$, що всі три користувачі вподобали фільм $\Phi1$, отже в них є спільний інтерес. Тепер завдання полягає в тому, щоб визначити які фільми будуть цікаві користувачеві $K3$. Візуалізація цього етапу на рис. 1.6.

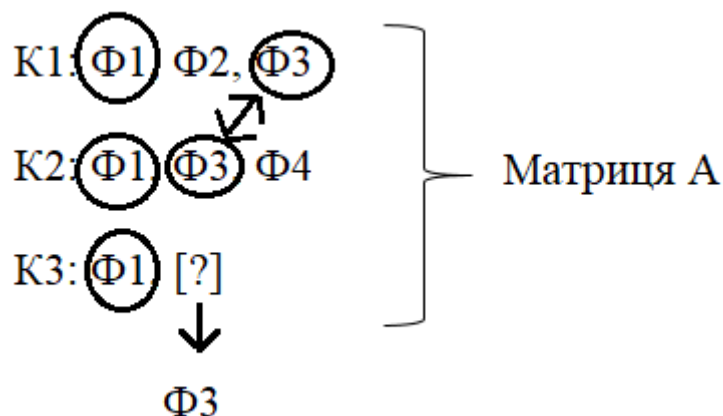


Рисунок 1.6 – схематичне зображення принципу роботи рекомендаційних систем з колаборативним фільтруванням на 2 етапі [4]

Виходячи з порівняння, можна побачити, що $\Phi1$ оцінили та вподобали всі три користувачі ($K1, K2, K3$). Також можна побачити, що $\Phi3$ оцінили та вподобали користувачі $K1$ та $K2$.

Оскільки K1 та K2 оцінили як Φ_1 , так і Φ_3 , існує висока ймовірність того, що Φ_3 також сподобається користувачеві K3, навіть незважаючи на інші інтереси користувачів K1 та K2.

В цьому і полягає головна ідея побудови рекомендаційних систем з колаборативним фільтруванням. Користувачі, які мали спільні інтереси в минулому, з великою долею вірогідності матимуть їх і в майбутньому.

Нижче, на рис. 1.7 наведено наочну різницю між принципами роботи рекомендаційних систем побудованих методом колаборативного фільтрування та на основі вмісту:

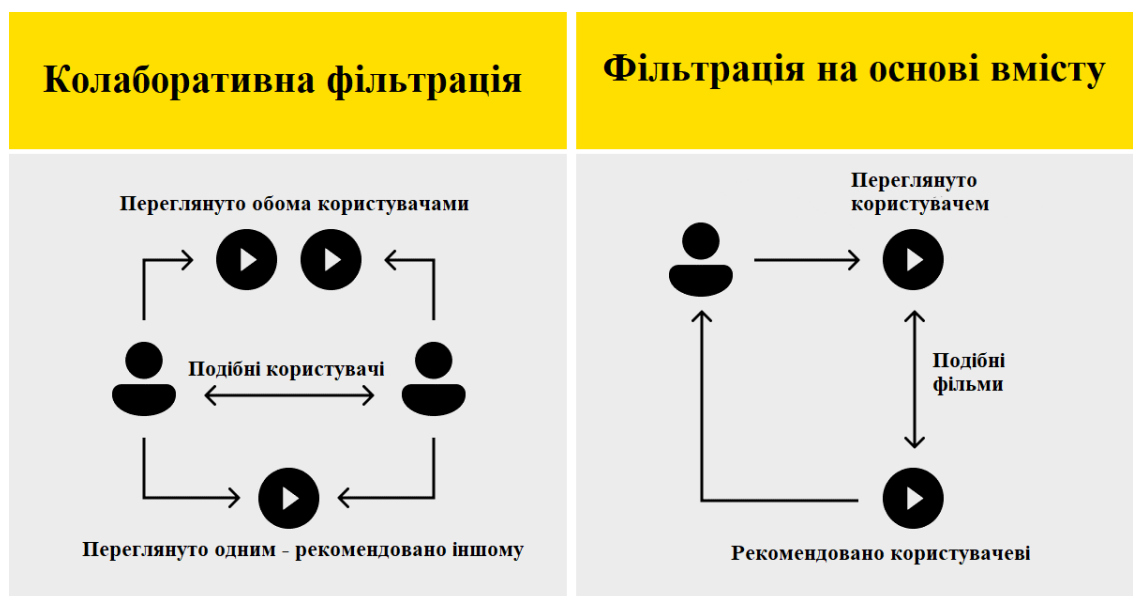


Рисунок 1.7 – Схематичне зображення різниці між принципами роботи рекомендаційних систем з колаборативним фільтруванням та фільтруванням на основі вмісту [21]

1.3.3 Гібридні системи

Гібридні системи, це ті системи, які поєднують різні існуючі моделі, такі як фільтрація на основі вмісту, колаборативна фільтрація або будь-яку іншу техніку рекомендації [12]. Гібридні системи можуть бути реалізовані шляхом окремого формування прогнозів на основі вмісту та на основі колаборативного

підходу, а потім їх комбінуванням, або шляхом об'єднання підходів у єдину модель. Наприклад, Netflix рекомендує фільми, порівнюючи звички перегляду та пошуку схожих користувачів (колаборативне фільтрування), а також пропонуючи фільми, що мають спільні характеристики з тими, що користувач оцінив високо (фільтрація на основі вмісту).

Розглянемо різні типи гібридних систем.

1. Пріоритетні системи використовують кілька різних моделей або методів рекомендацій, щоб вибрати той, який найточніше передбачає вподобання або інтереси користувача. Наприклад, якщо одна модель надає кращі результати для одного типу контенту (наприклад, фільми), а інша для іншого типу (наприклад, музика), система може вибирати модель в залежності від контексту запиту користувача;

2. Каскадні гібридні системи використовують послідовність моделей рекомендацій. Перша модель видає рекомендації, які стають вхідними даними для другої моделі, і так далі. Це дозволяє уточнювати рекомендації на кожному етапі і досягати кращої персоналізації.

3. Вибіркові гібридні системи вибирають одну конкретну техніку рекомендацій з набору доступних технік. Вибір може здійснюватися на основі характеристик користувача, контексту або відповідності контенту;

4. Мета-рівневі системи використовують результати однієї системи рекомендацій як вхід для іншої системи. Наприклад, результати колаборативної фільтрації можуть використовуватися для покращення роботи системи фільтрації на основі вмісту, і навпаки;

5. Комбінація ознак: гібридні системи, які використовують комбінацію ознак з різних джерел знань для формування одного домену рекомендацій. Це може включати не тільки характеристики вмісту або історії користування, а й додаткову інформацію, таку як демографічні дані чи попередні взаємодії користувача;

6. Доповнення ознак: ознаки одного джерела знань обчислюються або доповнюються для того, щоб зробити їх сумісними з іншими алгоритмами

рекомендацій. Наприклад, можливо, що інформація про вміст розширюється або адаптується для забезпечення кращих результатів колаборативної фільтрації.

Ці види гібридних систем використовуються переважно у сферах, де точність і різноманітність рекомендацій є критичними аспектами з точки зору бізнесу, такого як електронна комерція, стрімінгові послуги та соціальні мережі.

1.3.4 Системи на основі знань

Системи рекомендацій на основі знань використовують явну інформацію про асортимент товарів та вподобання клієнта. На основі цих знань система генерує відповідні рекомендації. Якщо жоден товар не відповідає всім вимогам, продукти, що задовольняють максимальний набір обмежень, ранжуються і відображаються. На відміну від інших підходів, цей метод не залежить від великих обсягів статистичних даних про товари або оцінок користувачів. Такий підхід дозволяє уникнути проблеми "холодного старту", оскільки рекомендації не залежать від бази оцінок користувачів. Системи рекомендацій на основі знань мають розмовний стиль, пропонуючи діалог, який ефективно веде користувача вниз по дереву властивостей продуктів.

Крім того, системи рекомендацій на основі знань можуть використовувати два основні підходи: на основі обмежень і на основі схожих випадків.

Системи на основі обмежень використовують явно визначений набір правил рекомендацій. Цей підхід передбачає, що користувач може вказати деякі обов'язкові вимоги щодо товарів, наприклад, місце розташування для нерухомості або розмір дисплея для електроніки. Система використовує ці обмеження для генерації рекомендацій.

Системи на основі схожих випадків засновані на інтелектуальному аналізі різноманітних мір подібності. Вони порівнюють вимоги користувача з атрибутами товарів у системі, щоб відібрати товари, які найбільше відповідають специфікаціям користувача. Цей підхід може використовувати алгоритми машинного навчання або експертні системи для визначення подібності між товарами [12].

Системи на основі знань особливо корисні у випадках, коли потрібно врахувати специфічні вимоги користувача, або коли необхідно працювати з товарами, які не мають багато інформації про їх реалізацію через те, що вони не часто продаються, такими як нерухомість.

1.4 Постановка задачі дослідження

Мета дипломної роботи – порівняння існуючих методів та алгоритмів побудови рекомендаційних систем і реалізація сервісу з рекомендації фільмів.

Об’єкт дослідження – процес генерації рекомендацій фільмів.

Предмет дослідження – методи машинного навчання та статистичного аналізу, їх використання для створення рекомендаційних систем, а також оцінювання якості згенерованих рекомендацій.

В ході виконання роботи проводиться дослідження та експериментальне порівняння різних методів генерації рекомендацій в рамках рекомендаційної системи на основі вмісту. Далі метод, що показав найкращі результати на основі метрик, інтегрується в розроблений інтерфейс для подальшого тестування та застосування.

Розроблена рекомендаційна система з рекомендації фільмів допоможе користувачам та сервісам отримувати персональні рекомендації на стрімінгових інтернет-платформах.

- надання миттєвих та персоналізованих рекомендацій;
- покращення бізнес-метрик сервісів, збільшуючи час користування ними;

- збільшення конверсії продажів послуг сервісів;
- покращення досвіду користувача у користуванні сервісом.

1.5 Висновки до розділу 1

У першому розділі виконано всебічний огляд рекомендаційних систем, розібрані існуючі типи, а також проаналізовано актуальність їх застосування. Зокрема були розглянуті рекомендаційні системи на основі вмісту, на основі колаборативного фільтрування, а також гібридні системи та системи на основі знань. Описано їх особливості та принципи роботи, що дозволяє зрозуміти їх важливість та роль у сучасних технологіях. Поставлена задача дослідження формує основу для подальших практичних кроків у розробці та вдосконаленні рекомендаційної системи.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1 Аналіз рекомендаційних систем відомих сервісів

Рекомендаційні системи можна назвати інноваційним сервісом. Система рекомендацій - це інструмент та техніка, яка надає користувачеві рекомендації на основі прогнозу, що рекомендований об'єкт йому сподобається. Тому їх використання можна знайти у сервісах та послугах всесвітньовідомих технологічних гігантів. Наприклад, корисні відео на YouTube, веб-серіали на Netflix, друзі на Facebook, книги на Amazon, будинки на Zillow, фільми на MovieLens тощо. Всі ці товари рекомендуються за допомогою рекомендаційних систем, які пропонується розглянути далі, щоб зрозуміти, які методики та алгоритми можна вважати такими, які зарекомендували себе у використанні таких великих компаній, які підході є актуальними в наш час, які оцінки роботи компанії використовують для контролю якості своїх рекомендацій.

Дедалі більше з'являється досліджень в області рекомендаційних систем, оскільки виникають нові методи та підходи для вдосконалення персоналізації. Тому розуміння сучасних тенденцій і методів у цій області є важливим аспектом будь-якого дипломного дослідження.

Рекомендаційні системи також впливають на величезне різноманіття інших галузей, таких як електронна комерція, медіа, соціальні мережі, наукові дослідження та інші, що підкреслює їх значення та актуальність в сучасному цифровому світі.

Таким чином, аналіз рекомендаційних систем відомих сервісів систем має великий потенціал для дослідження і розуміння передових методів та їх застосування у великих технологічних компаніях, що надає можливість для інновацій та вдосконалення існуючих практик.

Нижче пропонується розглянути 4 рекомендаційні системи всесвітньовідомих компаній: Airbnb, LinkedIn, Netflix, MovieLens.

2.1.1 Аналіз рекомендаційних систем Airbnb

Airbnb – це онлайн-платформа, що забезпечує можливість оренди житла користувачами. Власники нерухомості можуть здавати в оренду своє житло на короткий термін, а мандрівники можуть знаходити та бронювати житло через цю платформу. Airbnb надає широкий вибір місць для проживання у різних куточках світу, включаючи апартаменти, будинки, вілли та кімнати.

Для розбору цих неструктурованих даних великими масштабами Airbnb створила систему машинного навчання LATEX (Listing ATtribute EXtraction), яка може виокремлювати атрибути помешкань з неструктурованих текстових даних, таких як повідомлення гостей та відгуки, квитки обслуговування клієнтів та описи об'єктів. LATEX виконує це в наступні етапи: модуль визначення іменованих сутностей (NER) видобуває ключові фрази з неструктурованих текстових даних. А потім модуль відображення сутностей відображає ці ключові фрази на атрибути [15].

Ця система дозволяє Airbnb отримувати доступ до даних із великої кількості джерел, що допомагає краще розуміти інтереси гостей та вдосконалювати обслуговування.

Модуль визначення іменованих сутностей використовує текстову нейронну мережу і навчений, а також налаштований на людських позначених текстових даних з різних джерел в Airbnb. У навчальному наборі даних позначаємо кожен фразу, яка входить до однієї з наступних п'яти категорій: зручність, активність, подія та інші.

Компанія використовує модуль відображення сутностей, який використовує підхід навчання для відображення фраз, що характеризують атрибути дому. Для досягнення цього обчислюється косинусна відстань між

головною фразою та міткою атрибута в просторі векторного будування, який був налаштований. Найближче відображення вважається відповідним атрибутом і можна обчислити показник довіри для такого відображення.

Далі обчислюється як часто сутність згадується в кожному джерелі текстів (наприклад, повідомлення, відгуки, квитки обслуговування клієнтів), і агрегується нормалізована частота по всіх джерелах текстів, як показано на рис. 2.1. Атрибути дому, що мають багато згадок, вважаються більш важливими.

Рисунок 2.1 – Визначення ключових слів в рекомендаційній системі Airbnb [15]

За допомогою цієї системи Airbnb може отримати уявлення про те, що цікавить гостей, навіть виділяючи нові сутності, які ще не підтримуються. Масштабна інженерна система також дозволяє поліпшувати модель, підключаючи додаткові джерела даних і мови.

2.1.2 Аналіз рекомендаційних систем LinkedIn

LinkedIn - це професійна соціальна мережа, що забезпечує мільйонам користувачів у всьому світі можливість побудови професійних контактів, обміну знаннями та досвідом, а також знаходження робочих можливостей. На LinkedIn користувачі можуть створювати особисті профілі, де вони представляють своє резюме, професійний досвід, навички та досягнення. Вони можуть підключатися до інших користувачів, яких вони знають особисто чи

професійно, створювати і приєднуватися до груп, обмінюватися публікаціями та статтями, а також шукати роботу або наймати нових співробітників.

LinkedIn використовує класичну техніку колаборативної фільтрації, а саме техніку, що називається "онлайн-навчання" (online learning). Це метод машинного навчання, при якому модель неперервно оновлюється на основі нових даних, які надходять в режимі реального часу. Онлайн-навчання дозволяє моделі навчатися і адаптуватися до змінюваних умов без переривання роботи системи.

Основний компонент системи LinkedIn, що реалізує цей підхід, — це бібліотека машинного навчання Photon-ML, яка працює на основі Apache Spark. Apache Spark — це потужний фреймворк для обробки і аналізу великих обсягів даних в реальному часі. Photon-ML спеціально оптимізована для розподілених обчислень на Apache Spark і надає можливість використовувати онлайн-навчання для моделей машинного навчання.

Онлайн-навчання особливо корисне в ситуаціях, коли дані поступають в систему неперервно і потребують миттєвого аналізу та обробки. Наприклад, LinkedIn може використовувати цей підхід для підтримки рекомендаційних систем, аналізу взаємодій користувачів, виявлення аномалій або для персоналізованої реклами.

Таким чином, використання онлайн-навчання та Photon-ML дозволяє LinkedIn підтримувати високу ефективність своїх систем за рахунок швидкої адаптації до нових умов і даних, що поступають в реальному часі. На рис. 2.1.2 наведено візуалізацію роботи методу рекомендації цього сервісу, а саме в контексті персоналізації в режимі реального часу:

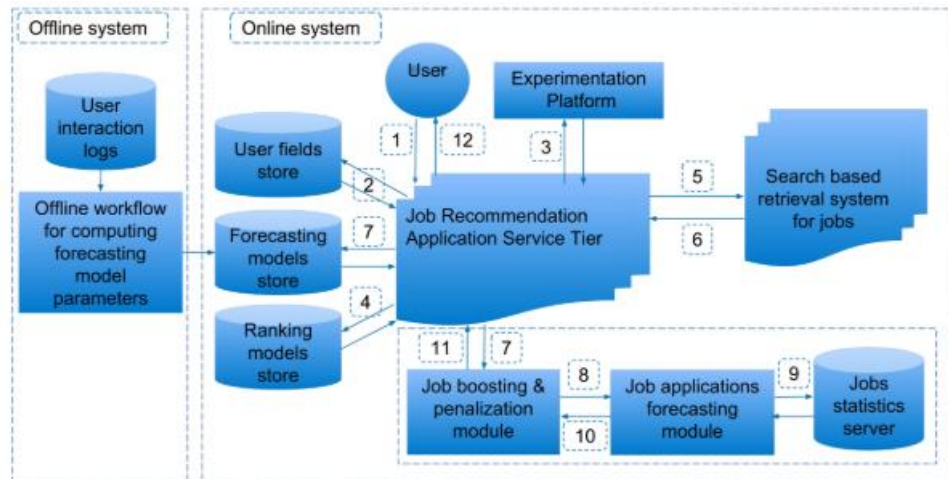


Рисунок 2.2 – Візуалізація роботи методу рекомендації сервісу LinkedIn [8]

2.1.3 Аналіз рекомендаційних систем Netflix

Netflix - це одна з найбільших платформ для стрімінгу відео у світі. Заснована у 1997 році, Netflix почав як платформа для оренди DVD, а потім перетворився на лідера ринку в онлайн-відеопослугах. Netflix пропонує широкий вибір фільмів, серіалів та оригінального контенту, який можна дивитися в режимі стрімінгу на різних пристроях, включаючи смартфони, планшети, телевізори та інше.

Netflix використовує сучасні технології машинного навчання і науки про дані для побудови своєї системи рекомендацій. Ця система відноситься до типу гібридних і призначена для того, щоб кожен користувач міг отримувати персоналізовані рекомендації щодо фільмів і шоу, які відповідають їхнім інтересам і вподобанням.

Розглянемо основні техніки, які застосовує Netflix.

1. **Матрична факторизація.** Це метод для розкладання матриці оцінок користувачів і фільмів на добуток двох матриць нижчого рангу. Це дозволяє прогнозувати оцінки, які користувачі можуть поставити фільмам, які вони ще не переглянули [17].

2. Reinforcement learning (навчання з підкріпленням). Використовується для оптимізації рекомендацій, враховуючи довгострокові наслідки. Наприклад, система може стимулювати користувачів дивитися нові або менш популярні фільми, що може покращити їхнє задоволення від сервісу.

3. Робота зі зваженими оцінками (weighted rating). Ця техніка враховує різний вплив оцінок користувачів на рекомендації. Наприклад, вагові коефіцієнти можуть використовуватися для підвищення вагомості оцінок від користувачів, які давно підписані на сервіс.

4. Колаборативне фільтрування і фільтрування змісту. Ці методи дозволяють аналізувати спільну взаємодію користувачів і вмісту для здійснення рекомендацій.

Netflix активно використовує ці техніки для покращення точності рекомендацій, залучаючи мільйони даних про оцінки від своїх передплатників. Це дозволяє платформі не тільки залучати і утримувати аудиторію, але і підвищувати задоволеність користувачів шляхом надання персоналізованого вмісту.

Також треба відмітити, що до 2019 року Netflix використовував рекомендаційну систему на основі вмісту, яка формувала рекомендації на основі даних про фільми, але з набуттям більшої потужності компанія перейшла на вищеописану методологію. На рис. 2.3 наведено візуалізацію алгоритму, який використовується в компанії:

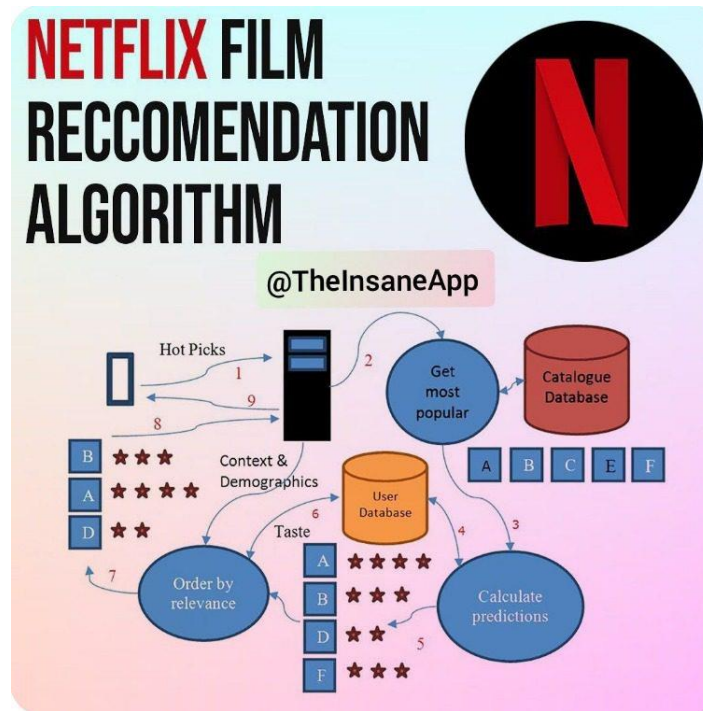


Рисунок 2.3 – Візуалізація алгоритму рекомендацій, який використовується в компанії Netflix [17]

2.1.4 Аналіз рекомендаційних систем MovieLens

MovieLens - це онлайн-платформа для рекомендацій фільмів, яка дозволяє користувачам отримувати персоналізовані рекомендації на основі їхніх інтересів і вподобань. Сервіс пропонує широкий вибір фільмів з різних жанрів і часових періодів, щоб задовольнити смаки будь-якого користувача.

Основними функціями MovieLens є підбір фільмів на основі історії переглядів користувача і аналізу схожих користувачів, що дозволяє створити індивідуальні рекомендації. Кожен фільм має оцінки користувачів, що дозволяє отримати об'єктивні відгуки та рейтинги, що враховуються при рекомендаціях [17].

MovieLens знаходить фільми, які сподобаються аудиторії, оцінює і потім рекомендує інші. На основі оцінок фільмів, MovieLens генерує персоналізовані прогнози для фільмів і досліджує різні аспекти персоналізації і технології фільтрації. В плані технологічного аспекту, відомо що сервіс

використовує класичну колаборативну фільтрацію, хоча цей факт легко передбачається з опису сервісу та способу на основі якого генеруються рекомендації. На рис. 2.4 наведено візуалізацію алгоритму рекомендації фільмів компанії MovieLens:

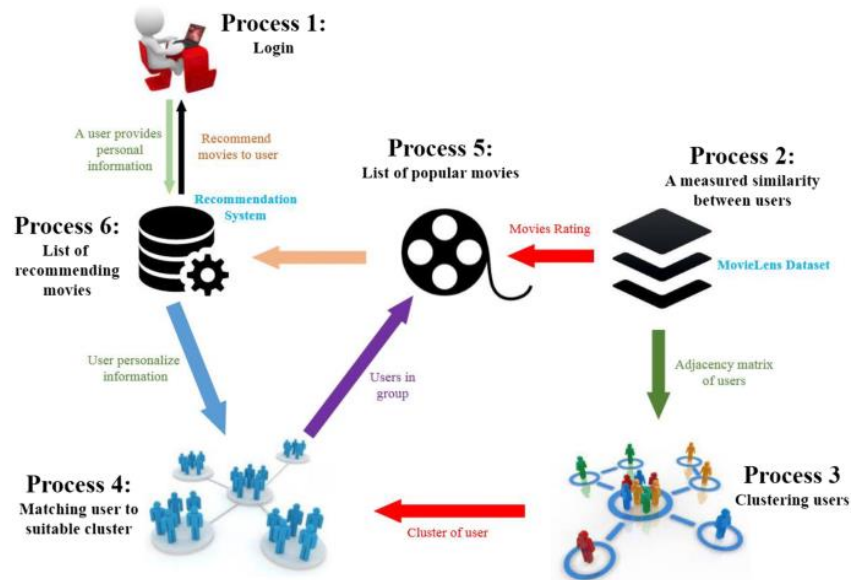


Рисунок 2.4 – Візуалізація алгоритму рекомендації фільмів для бізнес-моделі MovieLens [20]

2.1.5 Висновок порівняльного аналізу систем

На основі дослідження чотирьох великих рекомендаційних систем було виявлено, що вони не мають постійної вподобаної моделі рекомендаційної системи. Ці системи використовують різні підходи, такі як колаборативне фільтрування, гібридні техніки і рекомендації на основі контенту. Це свідчить про те, що вибір підходу залежить від конкретного контексту і завдань, що ставляться перед системою.

Більшість цих систем сильно покладаються на штучний інтелект та методи машинного навчання для аналізу великих обсягів даних і генерації персоналізованих рекомендацій. Вони використовують складні алгоритми для прогнозування і рекомендацій на основі різноманітних факторів, включаючи

історію перегляду, вподобання користувачів, схожість з іншими користувачами та властивості вмісту.

На основі проведеного аналізу можна зробити висновок, що при розробці програмного продукту варто орієнтуватися на досвід великих компаній і не копіювати їхні рішення, оскільки багато інформації про реальні методи є корпоративною таємницею. Проте вивчення і адаптація передових підходів допоможе створити конкурентоспроможну систему рекомендацій, яка задовольнить потреби індивідуальних користувачів.

2.2 Аналіз критеріїв якості для рекомендаційних систем

Рекомендаційні системи використовуються для прогнозування того, які предмети чи послуги можуть бути цікавими користувачам на основі їхніх попередніх взаємодій з платформою. Один з ключових аспектів при розробці рекомендаційних систем - це оцінка їхньої точності та якості. У цьому розділі розглянемо основні критерії якості, які використовуються для оцінки ефективності таких систем, зокрема Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Precision, Recall та F1-Score.

Mean Squared Error є однією з основних метрик для оцінки точності прогнозування рекомендаційних систем. Визначається вона як середнє значення квадрата різниці між фактичною і прогнозованою оцінками. Формула MSE (2.1):

$$MSE = \frac{1}{n} \sum_{i=1}^n (r_i - \hat{r}_i)^2, \quad (2.1)$$

де n – кількість елементів у вибірці; r_i – фактична оцінка для елемента i ; $\hat{r}_i$ – прогнозована оцінка для елемента i .

MSE вимірює середньоквадратичну помилку між прогнозованими і фактичними значеннями, тому менші значення MSE вказують на кращу точність моделі рекомендаційної системи.

Хороший результат MSE повинен бути близько до нуля, оскільки метрика вимірює квадрат відхилень між спостережуваними та передбаченими значеннями. Чим менше MSE, тим краще.

Root Mean Squared Error (2.2) є коренем квадратного з MSE і дозволяє отримати метрику в тих самих одиницях, що й оцінки:

(2.2)

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_i - \hat{r}_i)^2}$$

RMSE є більш наочним для прогнозних помилок, оскільки вона виражена у тих самих одиницях, що і фактичні оцінки. Чим менше значення RMSE, тим краще точність моделі.

Оптимальне значення RMSE також близьке до нуля, але в контексті порівняння моделей, там, де низькі значення RMSE вказують на кращі результати.

Mean Absolute Error (2.3) вимірює середню абсолютну різницю між фактичними і прогнозованими значеннями:

(2.3)

$$MAE = \frac{1}{n} \sum_{i=1}^n |r_i - \hat{r}_i|,$$

де: $|r_i - \hat{r}_i|$ - абсолютна різниця між фактичною і прогнозованою оцінками.

MAE є корисним показником для оцінки середньої величини помилок в прогнозах, незалежно від напрямку помилки.

MAE вимірює середнє абсолютне значення відхилень між спостережуваними та передбаченими значеннями. Чим менше MAE, тим краще.

Precision, Recall та F1-Score – ці метрики використовуються для оцінки класифікаційних моделей, зокрема в задачах, де важливо визначити правильність і повноту рекомендацій.

Precision (2.4) вимірює частку правильних рекомендацій серед усіх наданих рекомендацій:

(2.4)

$$Precision = \frac{TP}{TP+FP},$$

де TP (True Positives) – кількість правильно рекомендованих елементів;
FP (False Positives) – кількість неправильно рекомендованих елементів.

Для Precision бажане значення близько до 1.

Recall (2.5) вимірює частку правильно рекомендованих елементів серед усіх релевантних елементів:

(2.5)

$$Recall = \frac{TP}{TP + FN},$$

де FN (False Negatives) – кількість релевантних елементів, які не були рекомендовані.

Для Recall також бажане значення близько до 1.

F1-Score (2.6) є гармонійним середнім значенням Precision та Recall, що забезпечує баланс між цими двома метриками:

(2.6)

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Чим більше значення показника F1, тим краще.

Метрики MSE, RMSE та MAE є основними для оцінки точності прогнозування числових оцінок у рекомендаційних системах. Ці метрики допомагають визначити, наскільки близько прогнозовані значення відповідають фактичним оцінкам користувачів.

MSE є середнім значенням квадратів різниць між фактичними та прогнозованими оцінками. Ця метрика чутлива до великих помилок, оскільки квадрати різниць акцентують більше значення на великих відхиленнях. Це значення використовується для загального вимірювання точності моделей, де великі помилки є особливо важливими і повинні бути мінімізовані.

RMSE є квадратним коренем з MSE, що робить його більш інтерпретованим, оскільки він виражений в тих самих одиницях, що й фактичні оцінки. Він використовується для оцінки загальної точності моделей, де є потреба у відображенні помилок у зрозумілих одиницях виміру.

MAE є середнім значенням абсолютних різниць між фактичними та прогнозованими оцінками. Ця метрика є менш чутливою до великих помилок порівняно з MSE. Її можна використати для оцінки середньої величини помилок, що допомагає зрозуміти середнє відхилення прогнозів від реальних значень.

Ці три метрики дозволяють розробникам рекомендаційних систем оцінювати і вдосконалювати моделі на основі точності їх прогнозів. Використання MSE, RMSE та MAE допомагає виявити як великі, так і малі помилки у прогнозах та адаптувати моделі для більш точної рекомендації контенту.

Метрики Precision, Recall та F1-Score є критично важливими для оцінки якості класифікаційних моделей в рекомендаційних системах, особливо коли йдеться про визначення того, чи буде елемент рекомендований користувачу.

Precision вимірює частку правильних рекомендацій серед усіх наданих рекомендацій і використовується для оцінки якості рекомендацій, щоб мінімізувати кількість нерелевантних або помилкових рекомендацій. Висока точність означає, що серед рекомендованих елементів більшість є релевантними для користувача.

Recall вимірює частку правильно рекомендованих елементів серед усіх релевантних елементів і використовується для оцінки здатності системи охопити всі можливі релевантні рекомендації, що важливо для забезпечення повного задоволення користувача. Висока точність означає, що система змогла виявити більшість релевантних елементів.

F1-Score є гармонійним середнім Precision та Recall і забезпечує баланс між ними. Це значення використовується для загальної оцінки ефективності моделі, особливо в умовах, коли важливо забезпечити як високу точність, так і високу повноту. Він є особливо корисним, коли необхідно знайти компроміс між точністю і повнотою.

Метрики Precision, Recall та F1-Score дозволяють оцінити, наскільки добре рекомендаційна система знаходить релевантні елементи і наскільки вона здатна уникати нерелевантних рекомендацій. Це особливо важливо для підвищення задоволення користувачів і ефективності системи, оскільки правильне визначення релевантних рекомендацій безпосередньо впливає на користувацький досвід.

Метричні показники MSE, RMSE, MAE, Precision, Recall та F1-Score є важливими інструментами для аналізу та оцінки якості рекомендаційних систем. Вони дозволяють здійснювати кількісну оцінку того, наскільки добре модель прогнозує інтереси користувачів і надає релевантні рекомендації. Для досягнення кращих результатів у розробці рекомендаційної системи важливо

ретельно аналізувати та вдосконалювати модель за допомогою цих метрик, що забезпечить більш точні та ефективні рекомендації для користувачів.

2.3 Методи векторизації та їх аналіз

Згідно з метою та постановкою задачі буде розроблена рекомендаційна система на основі фільтрування вмісту. Основною частиною цього процесу є векторизація тексту, яка перетворює текстові дані у числовий формат, зручний для обробки алгоритмами машинного навчання.

Текстові дані, такі як статті, відгуки, або описи продуктів, є важливими джерелами інформації для багатьох застосувань, зокрема для рекомендаційних систем, аналізу настроїв, класифікації текстів та інших задач. Проте комп'ютери не можуть безпосередньо обробляти текст у його сирому вигляді. Векторизація тексту вирішує цю проблему шляхом перетворення тексту у числові вектори. Ці вектори зберігають інформацію про структуру та зміст тексту, дозволяючи алгоритмам машинного навчання аналізувати та робити висновки на основі цих даних.

Для цього існують різні інструменти для обробки природної мови – Natural Language Processing (NLP). Найбільш поширеними методами векторизації тексту є TF-IDF, CountVectorizer [5]. У цьому розділі пропонується розглянути кожен з цих інструментів детальніше.

Term frequency-inverse document frequency (TF-IDF) (частотність терміну - зворотна частотність документа) є технікою обробки природної мови, яка використовується для вимірювання важливості різних слів у реченні. Її алгоритм наведено на рис. 2.5. Вона усуває недоліки технік які не беруть до уваги частоту повторення слів в документі, і є корисною для класифікації тексту або для допомоги машинному навчанню розуміти слова.

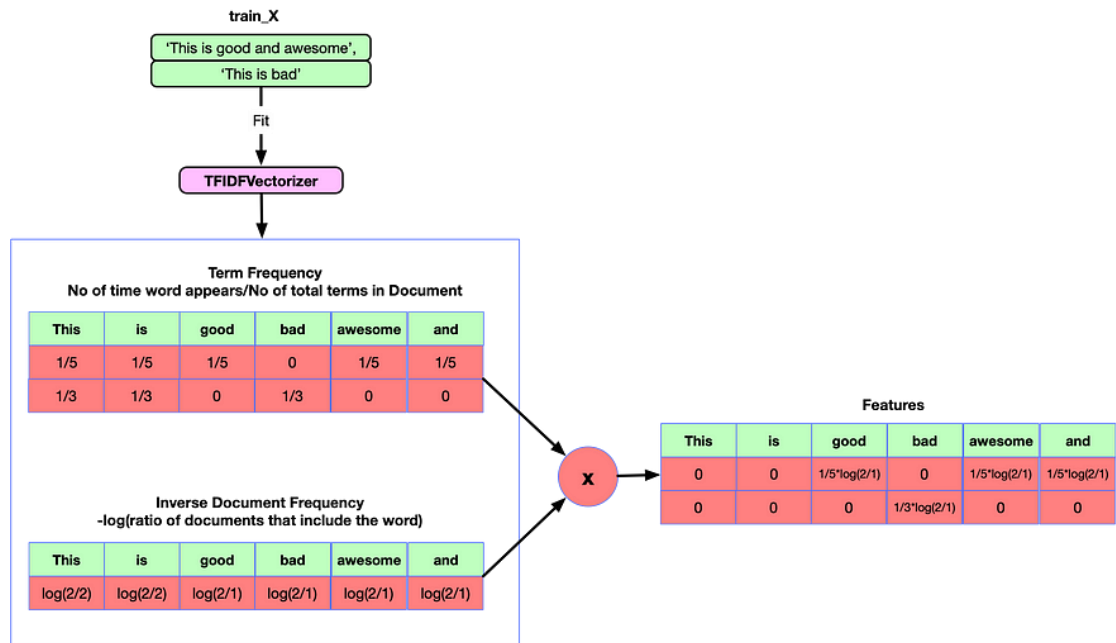


Рисунок 2.5 – Алгоритм TF-IDF [5]

TF-IDF є технікою обробки природної мови (NLP), яка використовується для оцінки важливості різних слів у реченні. Вона корисна для класифікації тексту та для допомоги моделі машинного навчання читати слова.

Нижче наведені терміни, які необхідно зрозуміти, щоб створити модель TF-IDF:

- t — term (термін, або слово);
- d — document (документ, або набір слів);
- N — кількість документів у корпусі;
- корпус — загальний набір документів.

Щоб пояснити що таке Term Frequency (TF), частота терміну потрібно вдатися до прикладу. Припустімо, є набір англійських текстових документів, і потрібно визначити, який документ найбільш релевантний до запиту "Natural Language Processing is valuable tool" [1]. Один із способів почати — це виключити документи, які не містять цих слів, але це все ще залишить занадто багато нерелевантних документів. Щоб далі їх розрізняти, можна підрахувати кількість появ кожного терміну в кожному документі. Кількість появ терміну в документі називається частотністю терміну. Формула TF (2.7):

(2.7)

$$TF(t, d) = \text{count of } t \text{ in } d / \text{number of words in } d$$

В той же час Document Frequency (DF), частотність документу вимірює важливість документа у всьому наборі корпусу. Це дуже схоже на TF. Єдина різниця в тому, що TF є лічильником частоти для терміну t у документі d , тоді як DF є кількістю появ терміну t у наборі документів N . Іншими словами, DF є кількістю документів, у яких присутнє слово.

Під час обчислення TF усі терміни вважаються однаково важливими. Однак певні терміни, такі як "is" "or" і "for" можуть з'являтися дуже часто, але мати мало значення. Потрібно зменшити вагу частих термінів, одночасно збільшуючи вагу рідкісних. Коли обчислюємо IDF, фактор зворотної частотності документа враховується, що зменшує вагу термінів, які часто зустрічаються у наборі документів, та збільшує вагу термінів, які рідко зустрічаються.

IDF є зворотним до частотності документа, що вимірює інформативності терміну t . Коли обчислюємо IDF, значення буде дуже низьким для найбільш частих слів, таких як стоп-слова "is" Це тому, що ці слова присутні майже у всіх документах, і формула (2.8) врахує низький пріоритет для таких слів:

(2.8)

$$IDF(t) = N/DF$$

Але ця формула не є остаточною, тому що є кілька інших проблем з IDF. Якщо у нас великий корпус, скажімо, 100.000.000 документів, значення IDF стає дуже великим. Щоб уникнути цього, береться логарифм від IDF. Також під час запиту, коли зустрічається слово, якого немає в словнику, значення DF буде 0. Оскільки не можна ділити на 0, згладжуємо значення, додаючи 1 до знаменника. Отримаємо остаточну формулу (2.9):

(2.9)

$$IDF(t) = \log\left(\frac{N}{DF + 1}\right)$$

Отже, TF-IDF — це міра, яка використовується для оцінки важливості слова в документі, колекції або корпусі. Існує багато різних варіантів TF-IDF, але наразі зосередимося на базовій версії (2.10):

(2.10)

$$TF - IDF(t, d) = tf(t, d) * \log\left(\frac{N}{DF + 1}\right)$$

CountVectorizer - це метод, який ще називають «мішок зі словами». Він перетворює колекцію текстових документів у числову матрицю підрахунку слів або токенів. Його алгоритм наведено на рис. 2.6. Цей клас має ряд параметрів, які можуть допомогти у завданнях передобробки тексту, таких як видалення стоп-слів, порогові значення підрахунку слів (наприклад, максимальні та мінімальні значення), обмеження словника, створення n-грам і багато іншого.

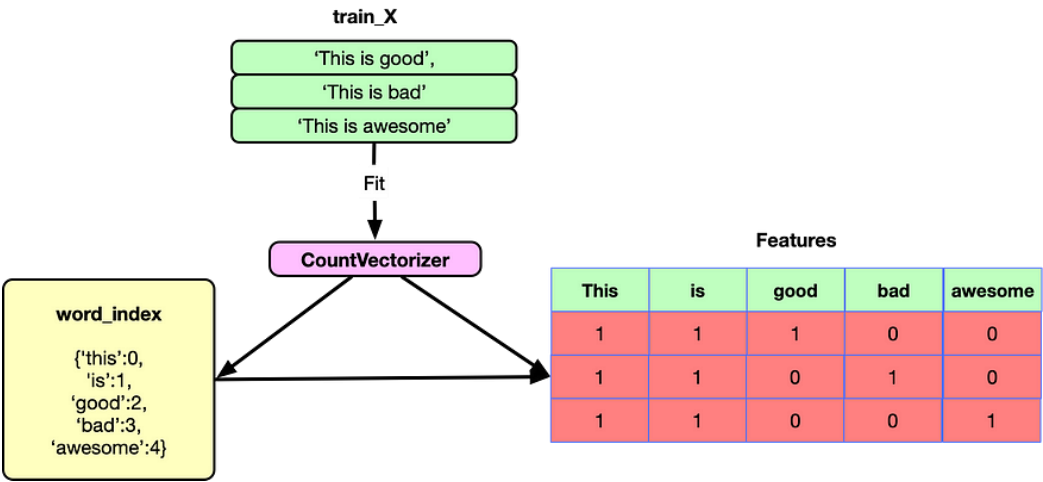


Рисунок 2.6 – Алгоритм CountVectorizer [5]

Розглянемо як цей метод працює на маленькому прикладі:

Нехай існує документ, який складається з трьох текстів: «One student helps two students», «Two students help three students», «Each student helps many other students at University».

Метод CountVectorizer створить матрицю в якій кожне слово буде відповідати колонкам, а кожен текст – рядку. Значенням кожної комірки матриці буде відповідати значенню кількості разів появи кожного слова в кожному тексті. Для нашого прикладу це матиме такий вигляд, як на рис. 2.7:

	at	each	four	help	helps	many	one	other	student	students	two	university
текст1	0	0	0	0	1	0	1	0	1	1	1	0
текст2	0	0	1	1	0	0	0	0	0	2	1	0
текст3	1	1	0	0	1	1	0	1	1	1	0	1

Рисунок 2.7 – Матриця методу CountVectorizer [5]

Отже, метод переводить всі літери слів до нижнього регістру, розміщує слова в колонках за алфавітом. Маємо 12 слів у трьох текстових наборах, отже і маємо розмірність матриці 12 на 3. Проте сам метод не зберігає слова у такому вигляді. Він пропонує кожному слову індекс, і матриця має виглядати як на рис. 2.8:

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	1	0	1	0	1	1	1	0
0	0	1	1	0	0	0	0	0	2	1	0
1	1	0	0	1	1	0	1	1	1	0	1

Рисунок 2.8 – Фінальна матриця методу CountVectorizer [5]

Отже, CountVectorizer - це простіший метод, який може працювати добре для деяких застосувань, але не завжди ефективний у виявленні

найважливішої інформації в тексті. З іншого боку, TF-IDF є кращим за CountVectorizer, оскільки він не лише фокусується на частоті слова в корпусі, але й надає важливість цих слів. Це дозволяє видаляти менш важливі слова для аналізу, що зменшує складність моделювання шляхом зменшення розмірності вхідних даних.

2.4 Методи машинного навчання

Алгоритми машинного навчання в контексті рекомендаційних систем, це алгоритми, що використовуються для фільтрації інформації та добування даних, для досягнення бажаних результатів. Важливо розуміти роботу методів фільтрації інформації, щоб вибрати правильний алгоритм для конкретного завдання в рекомендаційних системах [7].

2.4.1 Кластеризація K-Means

Кластеризація K-Means – це один із найпростіших підходів колаборативної фільтрації, який категоризує користувачів на основі їх інтересів. Зазвичай, якщо хтось хоче придбати товар, він запитує думку у того, хто вже купив цей продукт. Є висока ймовірність, що вплив теперішнього власника вплине на вподобання та смаки потенційного нового власника. Подібним чином алгоритм порівнює цікаві характеристики, які можуть бути пов'язані з індивідами, класифікованими у певну групу.

K-Means кластеризація використовує спільні інтереси серед користувачів, такі як вік, стать, час перегляду фільмів, історія попередніх переглядів тощо. Мета K-Means кластеризації полягає в тому, щоб згрупувати характеристики в кластери, які представляють характеристики групи. Якщо класифікація базується на віці, то ймовірна K-Means кластеризація використовуватиме методи кластеризації дітей, підлітків, молоді та дорослих.

Якщо клієнт потрапляє в будь-яку з цих вікових груп, фільми рекомендуються на основі того, що інші люди в цій віковій групі роблять. Якщо кластеризація залежить від віку, то чим ближче вік до центроїдного віку, тим краще рекомендація. Кроки класифікації включають вимірювання схожості між користувачем та характеристиками елементів, вибір сусідів, обчислення прогнозу та його рекомендацію.

Перший крок алгоритму полягає у знаходженні схожості у характеристиках користувачів, які новий користувач має з попередніми користувачами системи [7]. Алгоритм завжди має базові класифікації, де користувач може вводити дані, і прогнози можуть бути зроблені. Звичайні характеристики при пошуку схожостей – це вік, попередня історія та географічне розташування. Відстань від центроїдів може базуватися на кореляції Пірсона, косинусній схожості або на її коригованій версії. Обчислення схожостей може бути на основі елементів або користувачів. Обчислення на основі елементів знаходить схожості на основі характеристик фільмів, які подобались схожим людям. Якщо це обчислення на основі користувачів, то обчислення центроїдів базується на демографічних характеристиках користувача.

Обчислення схожостей між елементами або користувачами показано в рівнянні, що обчислює кореляцію між користувачем та елементом; воно обчислює близькість значення до центроїдного значення (2.11). Припускається, що два елементи $i \cap j$ є корельованими характеристиками (елементами або користувачами); значення $\bar{r}_i$ є центроїдною характеристикою, тоді як значення r_i є значенням нового користувача або новою характеристикою, що порівнюється за допомогою кореляції.

(2.11)

$$sim_{i,j} = \frac{\sum_{m \in i \cap j} r_{i,m} - \bar{r}_i r_{j,m} - \bar{r}_j}{\sqrt{\sum_{m \in i \cap j} r_{i,m} - \bar{r}_i^2} * \sqrt{\sum_{m \in i \cap j} r_{j,m} - \bar{r}_j^2}}$$

Наступний крок – вибір сусідів. Ключовими метриками є точність отримання та час виконання алгоритму. Для збільшення точності алгоритму потрібна велика кількість сусідів, що збільшує обчислювальний час алгоритму. Якщо потрібен менший обчислювальний час, точність буде знижена. Для досягнення балансу вибір може бути базований на пороговому значенні або на техніці топ-N. Порогова техніка буде запускати лише конкретну кількість (вибірку, що відповідає пороговому значенню) оцінок сусідів та прогнозувати, якщо цей поріг досягнутий. Наприклад, якщо популяція становить 1000, система запустить прогнозування з 100 вибірок та передбачить із цих 100 вибірок. У техніці топ-N запускаються лише топ-N схожостей, а не вся популяція сусідів. Наприклад, вона вибере лише топ-10 для пропозицій на основі найближчих сусідів, а не оцінюватиме всю популяцію.

Далі – обчислення прогнозу. Обчислення наступних прогнозів базується на найближчих сусідах, знайдених у системній базі даних. Прогноз або найближчий сусід до центроїда (K-Mean) обчислюється за формулою нижче. У наведеному рівнянні (2.12) K-Means представляється $\bar{r}_u$, тоді як кореляція іншої змінної з правого боку рівняння дає найближчого сусіда, обидва використовуються для створення рекомендації.

(2.12)

$$prediction_{u,i} = \frac{\sum_{n \in Neighbors} r_{n,i} - \bar{r}_n \ sim_{u,n}}{\sum_{n \in Neighbors} sim_{u,n}} + \bar{r}_u$$

Також алгоритм K-Means кластеризації має свої недоліки. Наприклад, проблема холодного старту – це проблема прогнозу, що виникає для нових користувачів систем. Є дуже мало інформації про користувача, тому системі важко зробити будь-які прогнозування, доки користувач не почне вводити деяку інформацію, яку можна корелювати, та робити рекомендації на основі характеристик користувача або попередніх елементів. Негативний вплив

полягає в тому, що точність системи значно знижується. Через цю проблему нові фільми не рекомендуються користувачам, або нові користувачі не знаходять те, що найкраще для них. Масштабованість також є однією з проблем, зазначених у виборі сусідів, необхідне балансування обчислювального часу та точності системи. Техніка фільтрації K-means є точною, коли база даних має невелику кількість фільмів для рекомендації або мало користувачів. Проте, зі збільшенням кількості користувачів та кількості фільмів, обчислювальний час або кількість елементів порогового значення збільшується, тому обчислювальний час збільшується. Щоб подолати цей недолік, обчислення та навчання алгоритму виконуються офлайн, з метою щоб коли системи повертаються онлайн, рекомендації робилися легко.

2.4.2 Метод головних компонентів K-Means

Метод головних компонентів K-Means – це техніка фільтрації фільмів на основі вмісту, яка покращує техніку кластеризації K-means. Основні компоненти у фільмі використовуються для класифікації фільмів перед тим, як можна робити рекомендації клієнтам. Алгоритм K-means обчислює близькість характеристики до центроїда, використовуючи відстань від середньої точки. Проте, метод головних компонентів створює матрицю коваріації для обчислення власних векторів та власних значень. Таким чином, він розширює масштабованість для знаходження кращих порівнянь для створення рекомендацій щодо фільмів. Для ілюстрації цього, припустимо, що алгоритм K-means обчислює схожість одного елемента за раз. Це означає, що обчислювальний час та точність компрометовані. Проте, використовуючи PCA, створюється матриця коваріації різних характеристик; таким чином, масштабованість збільшується та обчислюється швидше. Якщо є схожості, які потрапляють у матрицю, вони можуть бути легко знайдені, і обчислюється її

власний вектор. Рекомендації, близькі до такого власного вектора, потім створюються для рекомендації фільмів.

Для проведення методу головних компонентів спочатку потрібно сформувати дані. Це перший крок, дані формуються та структуруються у кортежі розміру $m \times n$. Можливі кортежі можуть залежати від характеристик користувача, характеристик фільму або комбінації характеристик користувача та фільму. Структура кортежів зображена на рис. 2.9:

Кортеж	Структура кортежу
A	$r_i \times u_i$
B	$r_i \times i_i$
C	$r_i \times u_i \times i_i$

Рисунок 2.9 – Структура кортежів [7]

де r_i – це рейтинг фільму; u_i – це характеристики користувача; i_i – це характеристики елементу (характеристики фільму). А та В дадуть 2D матрицю розміру $m \times n$, тоді як С дасть 3D матрицю розміру $m \times n \times o$.

Другим кроком обчислюється матриця коваріації розміру даних, сформованих на попередньому кроці. Далі йде обчислення власних векторів та власних значень: обчислена матриця коваріації буде квадратною матрицею розміру даних. Вона використовується для обчислення власних значень та власних векторів, які характеризують дані. Обчислені власні вектори сортуються у спадному порядку за власними значеннями.

2.4.3 Метод самоорганізуючих карт з аналізом головних компонентів

Самоорганізуючі карти (СОК) - це техніка, заснована на нейронних мережах, що використовує методи навчання без нагляду, тобто без втручання людини під час навчання. Вона є важливою для кластеризації даних без знання класових приналежностей у вхідних даних. Самоорганізуюча карта ознак (СОКО) відома тим, що виявляє ознаки, властиві певним елементам, що є важливим для рекомендаційних систем фільмів. СОК також використовує топологічно збережену мапу, що означає, що алгоритм зберігає відносні відстані між усіма точками в початковому наборі даних. Тому він ефективно досягає мети перетворення довільних вимірів у 1D або 2D дискретну мапу. Аналіз головних компонентів (АГК) інтегрується з СОК, оскільки АГК легше перетворює матриці, створені СОК, в власні вектори та власні значення для ранжування за значимістю.

Робота цього методу починається з ініціалізації – після отримання даних вибираються випадкові значення для ваг початкових векторів. Ваги векторів представляють нейрони в даних, і їхні значення також обчислюються.

Потім відбирається відомий зразок x з вхідного простору з відомою ймовірністю. Це активаційний шаблон, який застосовується до решітки. Цей шаблон відображає вимір x , пропорційний m -шаблону в новій решітці.

На основі цього найкраща відповідність знаходиться на кроці часу- n , використовуючи мінімальну евклідову відстань між центроїдом нейрона (2.13):

(2.13)

$$i(x) = \arg(\min |x - w_i|),$$

де $j = 1, 2, \dots i$.

Наступний крок – оновлення. Синаптична вага нейронів коригується за допомогою формули (2.14):

(2.14)

$$w_j(n+1) = w_j(n) + \eta(n)h_{ji(x)}(n)(x(n) - w_j(n)),$$

де $\eta(n)$ – це швидкість навчання; $h_{ji(x)}(n)$ – це функція сусідства нейрона $i(x)$.

Ці два параметри є динамічними для досягнення оптимальних результатів.

Застосування АГК відбувається після того, як синаптичні ваги отримані з мінімальної евклідової відстані за допомогою наведеної вище формули, процес АГК використовується для створення власних значень та власних векторів для подальшої обробки даних з метою отримання більш точних оцінок.

Після підбору схожостей здійснюються рекомендації, тобто прийняття рішення.

Основні переваги СОК, які роблять його гарним інструментом у рекомендаційних системах є відповідність щільності, тобто після введення даних у систему будь-які зміни у розподілі вхідних даних рівномірно відображаються у вихідній решітці, забезпечуючи гарне представлення областей з найвищою щільністю збігів та областей з меншою щільністю, та вибір ознак, тобто алгоритм СОК вибирає найкращі атрибути для нелінійного розподілу у вхідних даних, щоб ефективно зіставляти схожості з решітками.

2.4.4 Метод k-найближчих сусідів

Алгоритм k-найближчих сусідів (KNN) – це модель, яка класифікує точки даних на основі найбільш схожих на них точок. Вона використовує

тестові дані, щоб зробити "обґрунтоване припущення" щодо того, як слід класифікувати некласифіковану точку.

KNN є алгоритмом, який вважається як непараметричним, тобто алгоритм не робить жодних припущень щодо форми розподілу даних – модель повністю складається з наданих даних, а не виходить з того, що їх структура є нормальною, так і прикладом "лінивого" навчання, тобто метод майже не потребує навчання, через це всі навчальні дані також використовуються під час тестування, коли використовується KNN.

Цей алгоритм використовують такі компанії, як Netflix або Amazon, щоб рекомендувати різні фільми для перегляду або книги для покупки [9]. Компанії застосовують KNN до набору даних, зібраних про фільми, які дивилися користувачі, або книги, які купили на їхньому веб-сайті. Потім ці компанії введуть доступні дані як клієнта і порівнюють їх з іншими клієнтами, які дивилися подібні фільми або купували подібні книги. Ця точка даних буде класифікована як певний профіль на основі минулого за допомогою KNN. Фільми та книги, що рекомендуються, залежатимуть від того, як алгоритм класифікує цю точку даних.

На рис. 2.10 зображено як працює KNN, коли намагається класифікувати точку даних на основі заданого набору даних. Вона порівнюється з найближчими точками та класифікується на основі того, до яких точок вона найближча та найбільш схожа. Тут можна побачити, що точка X_j буде класифікована як або W_1 (червона), або W_3 (зелена) на основі її відстані від кожної групи точок.

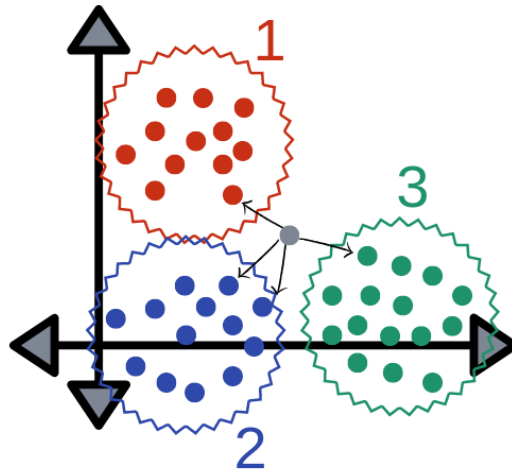


Рисунок 2.10 – Класифікація точки алгоритмом KNN [9]

При реалізації KNN першим кроком є перетворення точок даних у вектора ознак, або їх математичне значення. Алгоритм потім працює, знаходячи відстань між математичними значеннями цих точок. Найпоширеніший спосіб знайти цю відстань - використання евклідової відстані (2.15):

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (2.15)$$

KNN використовує цю формулу для обчислення відстані між кожною точкою даних та тестовими даними. Потім він знаходить ймовірність того, що ці точки схожі на тестові дані, та класифікує їх на основі того, які точки мають найвищі ймовірності.

2.5 Висновки до розділу 2

У цьому розділі було досліджено рекомендаційні системи, а саме їх типи та методи, які використовують у світових компаніях на прикладі компаній Airbnb, Netflix, LinkedIn, MovieLens. Було розглянуто алгоритми та проведено порівняльний аналіз методів векторизації тексту TF-IDF та CountVectorizer та методів машинного навчання – кластеризація K-Means, метод головних компонентів K-Means, метод самоорганізуючих карт з аналізом головних компонентів, алгоритм k-найближчих сусідів. Ці методи застосовуються в більшості моделей рекомендаційних систем. Також були детально проаналізовані особливості оцінки якості рекомендацій та відповідні метрики, включаючи їх роль, формули та значення показників.

РОЗДІЛ 3 РОЗРОБКА АРХІТЕКТУРИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ НА ПРИКЛАДІ СЕРВІСУ ВІДЕОСТРІМІНГУ

3.1 Вступ

Програмну розробку було проведено за допомогою мови Python. Вибір мови обґрунтовується підтримкою цією мовою бібліотек та інструментів, що полегшують та створюють комфортні умови для роботи з даними та їх аналізу. Також перевагу цій мові було надано через наявність великої кількості документації по роботі з машинним навчанням та статистичними методами та багато опублікованих робіт від спільноти розробників та інженерів-спеціалістів в галузі системного аналізу.

3.2 Розробка архітектури рекомендаційної системи на основі content-based моделі

Рекомендаційні системи використовують різні підходи для аналізу та прогнозування вподобань користувачів. Одним із ефективних методів є підхід з фільтрацією на основі вмісту, який базується на аналізі вмісту елементів, що оцінюються. У даній роботі розроблено рекомендаційну систему на основі вмісту з використанням методів машинного навчання та статистичного аналізу, проаналізовано ефективність впроваджених методів та побудовано інтерфейс для їх інтеграції.

Першим кроком є завантаження та об'єднання даних. Для цього використовуються два набори даних: `tmdb_5000_movies.csv` та `tmdb_5000_credits.csv`. Дані з обох файлів об'єднуються за загальною колонкою `title`, що дозволяє отримати повну інформацію про кожен фільм,

включаючи його опис, жанри, ключові слова, акторський склад та команду зйомок. Після об'єднання дані очищуються від пропущених значень.

Для створення моделі необхідно обробити текстові дані та перетворити їх у числові вектори. На цьому етапі виконується декілька ключових дій.

1. Перетворення даних: розбираються JSON-структури в колонках, таких як `genres`, `keywords`, `cast`, і `crew`, щоб отримати списки відповідних значень. Наприклад, для жанрів фільму витягуються назви жанрів, а для акторського складу — імена трьох головних акторів.

2. Очищення тексту: видаляються зайві пробіли та перетворюються всі символи в нижній регістр, що спрощує подальшу обробку тексту.

3. Стемінг: застосовується стемінг (перетворення слів до їх основних форм), що дозволяє зменшити кількість унікальних слів у тексті та покращити якість моделі.

4. Векторизація тексту: використовуються методи TF-IDF (Term Frequency-Inverse Document Frequency) та `CountVectorizer`. Ці методи дозволяють перетворити текстові дані у числові вектори, що необхідно для подальшого аналізу.

5. Обчислення подібності між фільмами: використовується метрика косинусної подібності. Косинусна подібність вимірює кут між векторами в багатовимірному просторі, що дозволяє оцінити, наскільки схожі тексти між собою незалежно від їхньої довжини. Це дозволяє визначити найбільш схожі фільми на основі їх опису, жанрів, ключових слів, акторського складу та команди зйомок.

Також застосовується алгоритм KNN (K-Nearest Neighbors) для знаходження найближчих сусідів. Цей метод дозволяє знаходити фільми, які найбільш схожі до заданого, використовуючи різні метрики відстані, такі як косинусна, евклідова та мангеттенська.

Для оцінки якості рекомендацій використовуються наступні метрики: Mean Squared Error (MSE) та Mean Absolute Error (MAE). Ці метрики використовуються для оцінки точності прогнозів. MSE вимірює середній

квадрат різниці між прогнозованими та фактичними значеннями, тоді як MAE вимірює середнє абсолютне відхилення.

На основі обчислених векторів та подібності формуються рекомендації для користувачів. Таким чином розроблена система надає рекомендації, використовуючи три різні підходи.

1. TF-IDF: Формує рекомендації на основі вагових векторів, отриманих методом TF-IDF.

2. CountVectorizer: Використовує частотні вектори для формування рекомендацій.

3. KNN: Використовує алгоритм K-Nearest Neighbors для знаходження найближчих сусідів та формування рекомендацій.

Розроблена система надає користувачеві можливість отримати рекомендації для заданого фільму. Наприклад, для фільму "Skyfall" будуть надані рекомендації з використанням кожного з трьох підходів.

Архітектуру програмного продукту можна описати блок-схемою на рис. 3.1. Блок-схема ілюструє основні етапи процесу: завантаження даних, обробка, векторизація тексту, обчислення подібності та формування рекомендацій для користувачів.

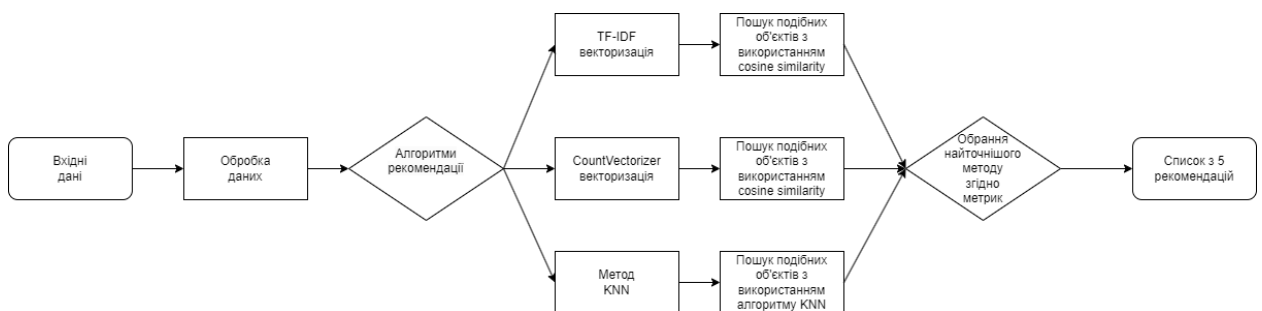


Рисунок 3.1 – Блок-схема архітектури програмного продукту

Узагальнюючи, розроблена архітектура рекомендаційної системи з фільтруванням на основі вмісту дозволяє ефективно аналізувати та рекомендувати фільми, враховуючи різні аспекти їхнього вмісту. Система

використовує методи TF-IDF, CountVectorizer та KNN для формування рекомендацій, що дозволяє забезпечити високу якість та точність прогнозів.

3.3 Огляд використаних бібліотек

При розробці рекомендаційної системи використовувались наступні бібліотеки: pandas, numpy, nltk, sklearn, scipy. Кожна з них відіграє важливу роль в процесі побудови, навчання та оцінки системи. Нижче наведено огляд цих бібліотек та їх застосування в контексті рекомендаційних систем.

Pandas – це потужна та гнучка бібліотека для роботи з даними, яка забезпечує високоефективні, прості у використанні структури даних та аналіз даних на мові Python. Основні можливості: структури даних. Pandas надає дві основні структури даних: DataFrame (двовимірна таблиця) і Series (одновимірний масив). Ці структури дозволяють зручно зберігати та маніпулювати даними.

Операції з даними. Бібліотека дозволяє легко здійснювати операції з даними, такі як сортування, фільтрація, об'єднання, агрегація та індексація.

Зчитування та запис даних. Pandas підтримує різні формати даних, включаючи CSV, Excel, SQL, JSON та інші, що робить її універсальним інструментом для роботи з даними.

Pandas використовується для зчитування даних з файлів, їх очищення та підготовки до подальшого аналізу.

Очищення даних та робота з відсутніми значеннями. Бібліотека надає інструменти для виявлення та обробки відсутніх значень, що є критично важливим для створення якісних моделей.

Аналіз та підготовка даних для побудови моделей. Pandas допомагає аналізувати розподіл даних, виконувати попередню обробку та трансформацію даних, що покращує якість рекомендаційної системи.

Numpy – це фундаментальна бібліотека для наукових обчислень на Python, яка забезпечує підтримку великих багатовимірних масивів та матриць, а також велику кількість математичних функцій для роботи з ними.

Numpy надає багатовимірні масиви (ndarray), які є ефективними структурами для зберігання та маніпуляції числовими даними.

Бібліотека дозволяє виконувати векторизовані операції над масивами, що значно підвищує швидкість обчислень порівняно з традиційними циклічними підходами.

Numpy включає великий набір математичних функцій для обчислення статистик, алгебраїчних операцій, генерації випадкових чисел тощо.

Numpy використовується для ефективного зберігання та обробки великих обсягів даних, таких як рейтинги користувачів або параметри моделей.

Використання numpy дозволяє значно прискорити виконання алгоритмів, таких як обчислення матричних факторизацій або подібностей.

Реалізація основних алгоритмів обробки даних. numpy забезпечує базові функції для реалізації алгоритмів машинного навчання та аналізу даних.

Nltk (Natural Language Toolkit) – це бібліотека для роботи з людською мовою, яка надає інструменти для обробки та аналізу текстових даних.

Nltk дозволяє розбивати текст на менші компоненти, такі як слова чи речення.

Стемінг та лемматизація дозволяють зводити слова до їхніх базових форм, що допомагає зменшити варіативність текстових даних. Також Nltk надає інструменти для аналізу граматичної структури тексту, що може бути корисним для складніших завдань обробки природної мови.

Використовується для аналізу текстових даних, таких як рецензії на фільми чи коментарі користувачів, що можуть бути важливими для формування рекомендацій.

Nltk дозволяє витягувати ключові слова та поняття з тексту, що допомагає в удосконаленні рекомендацій. Допомогає автоматично витягувати

релевантну інформацію з великих обсягів текстових даних, що може використовуватись для покращення моделей рекомендацій.

Scikit-learn (sklearn) – це бібліотека машинного навчання, яка надає простий та ефективний інструментарій для побудови та оцінки моделей машинного навчання.

Sklearn містить реалізації численних алгоритмів машинного навчання, включаючи регресію, класифікацію, кластеризацію та зменшення розмірності.

Бібліотека надає інструменти для нормалізації, масштабування та трансформації даних, що є важливим кроком у підготовці даних для моделей. Sklearn включає методи для крос-валідації та оцінки якості моделей за допомогою різних метрик, таких як точність, повнота та F1-міра. Використовується для побудови моделей машинного навчання, які прогнозують рейтинги або визначають, який контент рекомендувати. Sklearn надає алгоритми, які використовуються для знаходження схожих користувачів або елементів, що є основою багатьох рекомендаційних систем.

Оцінка якості моделей та їх точності. Бібліотека дозволяє оцінювати ефективність моделей, що допомагає вибрати найкращі підходи для конкретної задачі рекомендацій.

Scipy – це бібліотека для наукових та технічних обчислень, яка розширює функціональність numpy і надає великий набір функцій для роботи з математичними та статистичними задачами.

Scipy включає функції для роботи з матрицями, розв'язання систем лінійних рівнянь та обчислення власних значень. Містить методи для оптимізації функцій, що є корисним при налаштуванні параметрів моделей. Scipy надає широкий набір статистичних функцій, включаючи розподіли, перевірки.

Scipy використовується для виконання високоточних обчислень, необхідних для реалізації та оптимізації моделей. Бібліотека допомагає в статистичному аналізі даних, що є важливим для розуміння та інтерпретації

результатів моделювання. Scipy дозволяє застосовувати різноманітні статистичні методи для оцінки якості рекомендацій та покращення моделей.

Використання бібліотек `pandas`, `numpy`, `nlTK`, `sklearn` та `scipy` забезпечує потужний інструментарій для розробки рекомендаційних систем. Вони дозволяють ефективно обробляти та аналізувати дані, будувати моделі машинного навчання та оцінювати їхню якість, що є ключовим для створення надійної та ефективної системи рекомендацій.

3.4 Опис вхідних даних

Розроблена рекомендаційна система буде генерує рекомендації на основі даних, опрацьованих алгоритмом. Датасет сформований на основі даних, наданих сервісом The Movie Data Base (TMDB) та отриманий з публічного веб-сайту Kaggle [19]. Сервіс TMDB надав вільний доступ до даних про 5000 найпопулярніших фільмів у форматі двох датасетів формату CSV.

Перший датасет містить інформацію про вміст фільмів, описуючи титри фільмів містить такі поля як: ідентифікаційний номер фільму, назва фільму, акторський склад фільму, та відомості про знімальну команду, яка працювала над фільмом.

Другий датасет містить інформацію про вміст фільмів, описуючи наступну інформацію: ідентифікатор фільмів, опис фільму, ключові слова, мова фільму тощо.

Як бачимо, датасети мають всю необхідну інформацію щодо фільмів, на основі якої можна будувати метадані для рекомендаційної системи на основі вмісту.

Дуже важливо, щоб датасети мали спільне поле, за яким їх можна поєднати та використовувати як один. Що і було зроблено в роботі, датасети

були об'єднані за спільним полем назва фільму і збережено дані у новий датасет.

Звісно, такий вид даних має недолік – дані не мають автоматичного оновлення і рекомендації можуть бути застарілими. Проте це можна покращити, є вирішення і цієї проблеми – якщо буде існувати необхідність оновлювати датасет автоматично, то можна створити скрипт, який буде автоматично завантажувати потрібну інформацію про фільми з сервісу TMDb, використовуючи персональний API ключ. Проводити завантаження можна лише раз на тиждень – щочетверга, коли відбуваються прем'єри фільмів і в базі з'являється нова інформація про них.

За допомогою команди `.info()` бібліотеки Pandas було отримано коротку інформацію щодо типів даних та їх значення. На рис. 3.2 наведено результат виконання цього запиту:

```
Data columns (total 23 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   budget                                4809 non-null  int64
1   genres                                4809 non-null  object
2   homepage                              1713 non-null  object
3   id                                    4809 non-null  int64
4   keywords                              4809 non-null  object
5   original_language                     4809 non-null  object
6   original_title                         4809 non-null  object
7   overview                              4806 non-null  object
8   popularity                            4809 non-null  float64
9   production_companies                  4809 non-null  object
10  production_countries                  4809 non-null  object
11  release_date                          4808 non-null  object
12  revenue                                4809 non-null  int64
13  runtime                              4807 non-null  float64
14  spoken_languages                      4809 non-null  object
15  status                                4809 non-null  object
16  tagline                              3965 non-null  object
17  title                                4809 non-null  object
18  vote_average                          4809 non-null  float64
19  vote_count                            4809 non-null  int64
20  movie_id                             4809 non-null  int64
21  cast                                  4809 non-null  object
22  crew                                  4809 non-null  object
dtypes: float64(3), int64(5), object(15)
```

Рисунок 3.2 – Коротка інформація щодо типів даних та їх значень у датасеті

Як бачимо, замість заявлених 5000 фільмів отримали лише 4809, проте і цієї кількості буде достатньо для формування рекомендації.

3.5 Етапи створення реалізованого продукту

В цьому розділі буде описано яким чином були застосовані методи, які були розглянуті в попередніх розділах для створення рекомендаційної системи з фільтруванням на основі змісту з використанням векторної моделі. Далі описано процес застосування методів, розглянутих в попередніх розділах для створення рекомендаційної системи з фільтруванням на основі змісту з використанням векторної моделі. Використані методи векторизації TF-IDF та CountVectorizer та метод машинного навчання KNN, результати роботи яких буде порівняно між собою, і на основі методів з кращим показником буде зроблена фінальна рекомендація.

Також описано процес фільтрування на основі змісту для надання користувачам релевантних рекомендацій на основі векторних моделей представлення текстової інформації, які дозволяють зіставляти об'єкти за їх змістовими характеристиками.

3.5.1 Обробка даних

Отже, як зазначено у пункті 3.3, маємо статичний датасет, ціллю якого є створення та тестування моделі. Як вже було зазначено за допомогою базових команд бібліотеки Pandas було поєднано інформацію з двох датасетів у один за спільним полем «title», що містить в собі назву фільмів. Також за допомогою базових інструментів цієї ж бібліотеки було видалено записи які містили нульові значення та була зроблена перевірка на дублікати – жодних випадків дублювання даних не було виявлено. Такий датасет можна вважати готовим для роботи.

Після отриманого датасету, про який йшла мова в минулому пункті, маємо датасет з наступними полями, які зображено на рис. 3.2.

Проте наша ціль – створити датасет для рекомендаційної системи на основі фільтрації на основі вмісту. Тож датасет має виглядати наступним чином, тобто мати наступні поля: ідентифікатор фільму, назва, вміст. Якщо поля ідентифікатор фільму, назва не будуть зазнавати змін, то поле вміст буде розроблятися з 0 на основі даних які вже маємо.

Для того щоб розробити поле вміст було зроблено наступне:

- у полі знімальна команда (crew) було прибрано інформацію про всіх членів команди, окрім режисера фільму;
- у полі опис фільму (overview) всі речення було перетворено на набір слів, для того щоб дані поля можна було аналізувати за найчастіше згадуваними словами;
- поля жанрів (genres), опису (overview), ключових слів (keywords), списку акторів (cast), знімальна команда (crew) було об'єднано у єдине поле вміст (tags).

Зберігаємо це в новий датасет, перші 4 строки якого зображено на рис. 3.3:

new_df			
	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...

Рисунок 3.3 – Датасет зі створеним полем “tags”

Тепер роботу з датасетом завершено – в такому форматі файл є готовим до обробки його методами створення рекомендації. Проте саме поле tags може не уникнути розбіжностей між однаковими словами у різних формах і тим самим підвищити точність рекомендації. Для цього у наступному пункті буде відбуватися робота з даними цього поля.

3.5.2 Обробка метаінформації

Зараз в полі метаінформації кожного фільму, який у нас наведено у вигляді поля `tags`, міститься текст, всі слова якого поділені комами. Для приведення інформації до нормального вигляду було проведено наступні операції: символи поля `tags` повинні бути переведені до нижнього регістру, слова мають бути відокремленні пробілами за допомогою двох базових команд вибраної мови програмування, які зображені на рис. 3.4:

```
new_df['tags'] = new_df['tags'].apply(lambda x: " ".join(x))
new_df['tags'] = new_df['tags'].apply(lambda x:x.lower())
```

Рисунок 3.4 – Базові команди для переведення слів до нижнього регістру та відокремлення їх пробілами

Після того як всі символи поля `tags` буде переведено до нижнього регістру, датасет можна вважати готовим до векторизації, після того як буде оброблено інформацію у полі `tags` таким чином, щоб уникнути розбіжностей між однаковими словами у різних формах і тим самим підвищити точність рекомендації. Тож зараз поле метаінформації виглядає як зображено на рис. 3.5:

```
0      in the 22nd century, a paraplegic marine is di...
1      captain barbossa, long believed to be dead, ha...
2      a cryptic message from bond's past sends him o...
3      following the death of district attorney harve...
4      john carter is a war-weary, former military ca...
...
4804    el mariachi just wants to play his guitar and ...
4805    a newlywed couple's honeymoon is upended by th...
4806    "signed, sealed, delivered" introduces a dedic...
4807    when ambitious new york attorney sam is sent t...
4808    ever since the second grade when he first saw ...
```

Рисунок 3.5 – Вигляд поля метаінформації до використання методу PorterStemming

А за допомогою методу бібліотеки nltk PorterStemming буде виглядати як на рис. 3.6:

```

0      in the 22nd century, a parapleg marin is dispa...
1      captain barbossa, long believ to be dead, ha c...
2      a cryptic messag from bond' past send him on a...
3      follow the death of district attorney harvey d...
4      john carter is a war-weary, former militari ca...
...
4804   el mariachi just want to play hi guitar and ca...
4805   a newlyw couple' honeymoon is upend by the arr...
4806   "signed, sealed, delivered" introduc a dedic q...
4807   when ambiti new york attorney sam is sent to s...
4808   ever sinc the second grade when he first saw h...

```

Рисунок 3.6 – Вигляд поля метаданих після використання методу PorterStemming

Алгоритм Porter Stemmer, розроблений Мартіном Портером у 1980 році, є одним з найвідоміших і найчастіше використовуваних алгоритмів стемінгу для англійської мови. Його основна мета - звести різні форми слова до його основної форми, званої "стемом" [14]. Це досягається за допомогою видалення суфіксів та інших закінчень зі слів, що є дуже корисним у сфері інформаційного пошуку (Information Retrieval, IR). Цей процес зменшує кількість термінів, які зберігаються в системі IR, що буде корисно як з точки зору простору, так і часу обробки.

В цьому алгоритмі Приголосною є літера, яка не є голосною, а також літера "Y", якщо вона передуює приголосній. Таким чином, у слові "TOY" приголосними є "T" і "Y", а в слові "SYZYGY" - "S", "Z" та "G".

Якщо літера не є приголосною, то вона є голосною.

Приголосна буде позначатися як c, а голосна як v.

Список з однієї або більше послідовних приголосних (ccc...) буде позначатися як C, а список з однієї або більше послідовних голосних (vvv...)

буде позначатися як V. Будь-яке слово або частина слова, таким чином, мають одну з чотирьох форм, наведених нижче [14].

CVCV ... C → collection, management

CVCV ... V → conclude, revise

VCVC ... C → entertainment, illumination

VCVC ... V → illustrate, abundance

Тож ці 4 форми можна привести в одну загальну (3.1), де квадратні дужки позначають довільну наявність приголосних або голосних:

(3.1)

$$[C]VCVC...[V]$$

Якщо $(VC)^m$ позначає повторення VC m разів. Таким чином, вище наведений вираз можна записати у такому вигляді (3.2):

(3.2)

$$[C](VC)^m[V]$$

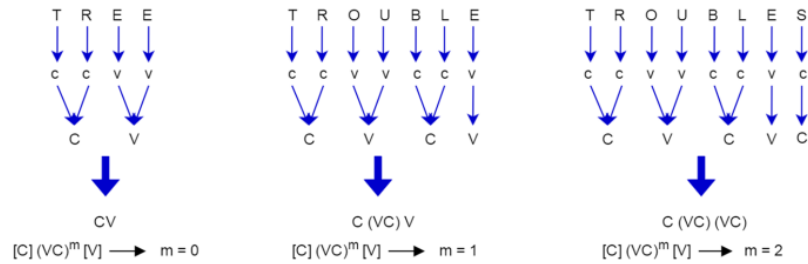
Значення m, знайдене у вище наведеному виразі, називається мірою будь-якого слова або його частини, коли воно представлене у формі (3.2). Ось кілька прикладів для різних значень m:

m=0 → TREE, TR, EE, Y, BY

m=1 → TROUBLE, OATS, TREES, IVY

m=2 → TROUBLES, PRIVATE, OATEN, ROBBERY

На рис. 3.7 наведено візуалізацію цього алгоритму.



Porter Stemming Algorithm

SSES	→	SS	(m>0) ATIONAL	→	ATE
IES	→	I	(m>0) TIONAL	→	TION
SS	→	SS	(m>0) ENCI	→	ENCE
S	→		(m>0)	→	ANCE

Рисунок 3.7 – Візуалізація алгоритму Porter Stemming [14]

Застосування цього алгоритму у коді зображено на рис. 3.8:

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
✓ 0.0s

def stem(text):
    y= []

    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)
```

Рисунок 3.8 – Застосування алгоритму Porter Stemming у коді

Підібрані бібліотеки дозволяють використовувати великий обсяг роботи у маленькому функціоналі, що оптимізує як роботу коду, так і сприйняття його розробниками.

Отже, було отримано стандартизоване поле метаданих у готовому датасеті. Можна вважати, що всі етапи підготовки даних та датасету виконані, і можна приступати до етапу створення моделі рекомендації.

3.5.3 Векторизація тексту

Векторизація тексту є ключовим етапом у створенні рекомендаційної системи на основі контенту. Векторизація перетворює текстову інформацію в числові вектори, які можуть бути оброблені алгоритмами машинного навчання для визначення схожості між фільмами. У нашій системі використовуємо два основні методи векторизації: TF-IDF (Term Frequency-Inverse Document Frequency) та CountVectorizer. Кожен з цих методів має свої переваги та застосовується для різних цілей.

Метод TF-IDF допомагає визначити важливість термінів у документі з урахуванням їх частоти у всьому наборі документів. Процедура векторизації за допомогою TF-IDF включає наступні кроки:

Використання бібліотеки `scikit-learn` для створення об'єкта `TfidfVectorizer` з параметрами, що включають максимальну кількість характеристик (5000) та виключення стоп-слів (таких як "the", "and", тощо).

Стоп-слова - це слова, які часто зустрічаються в текстах, але не несуть суттєвого значення для аналізу даних або побудови моделей машинного навчання. Їх використання в процесі векторизації тексту може призвести до збільшення розмірності векторів без додаткової інформативності. Тому, у більшості випадків, стоп-слова видаляють із текстових даних перед обробкою.

Важливість видалення стоп-слів полягає у можливості зменшення розмірності даних: видалення стоп-слів зменшує кількість слів у тексті, що веде до зменшення розмірності матриці термін-документ (TF-IDF або CountVectorizer). Це полегшує обробку даних та покращує продуктивність алгоритмів машинного навчання.

Стоп-слова не несуть значущої інформації для контексту або змісту документів. Їх видалення допомагає зосередитись на більш інформативних словах, що покращує точність та релевантність рекомендаційних систем та інших текстових моделей.

Стоп-слова можуть діяти як шум в текстових даних [1]. Видалення цього шуму допомагає отримати більш чисті та точні результати.

Стоп-слова можуть різнитися залежно від мови. У англійській мові прикладами стоп-слів є:

Займенники: I, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves

Прийменники: at, on, in, with, about, against, between, into, through

Артикли: the, a, an

Сполучники: and, or, but, because, so, if, while, although

Інші частки: is, am, are, was, were, be, been, being, have, has, had, do, does, did, will, would, shall, should, can, could, may, might, must, ought

При векторизації тексту за допомогою TF-IDF або CountVectorizer, стоп-слова автоматично виключаються за допомогою параметра `stop_words='english'`.

Застосування TF-IDF векторизатора до підготовлених текстових даних для створення матриці TF-IDF векторів. Кожен рядок цієї матриці представляє фільм, а кожен стовпець - значення TF-IDF для певного терміну. Реалізацію цих кроків в коді зображено на рис. 3.9:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# ініціалізація TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')

✓ 0.0s

vectors = tfidf.fit_transform(new_df['tags']).toarray()
```

Рисунок 3.9 – Застосування TF-IDF векторизатора

Цей процес дозволяє представити текстові дані у вигляді числових векторів, що є необхідним кроком для обчислення схожості між фільмами.

Для методу CountVectorizer всі кроки є аналогічними, а реалізацію цих кроків в коді зображено на рис. 3.10:

```

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000, stop_words='english')
✓ 0.0s

vectors2 = cv.fit_transform(new_df['tags']).toarray()
✓ 13.7s

```

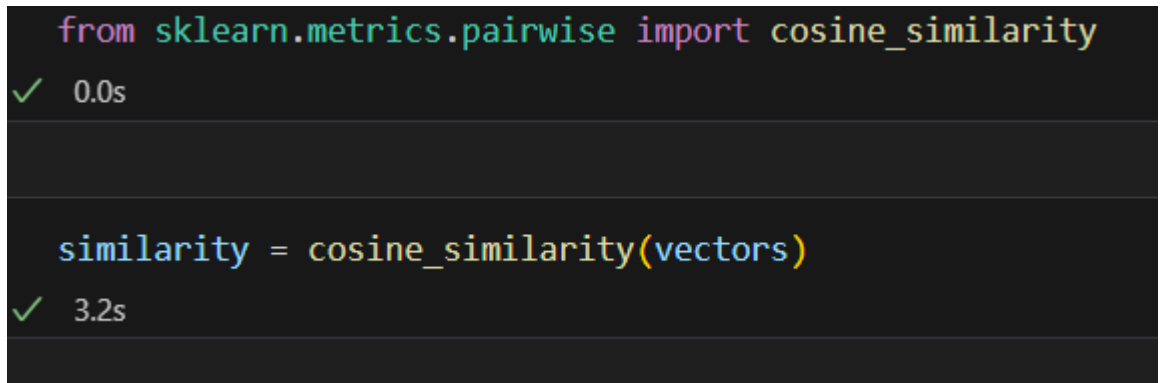
Рисунок 3.10 – Застосування векторизатора CountVectorizer

3.5.4 Функції рекомендації

Функції рекомендації є основним компонентом нашої системи, що відповідає за вибір та виведення фільмів, які найбільше відповідають вподобанням користувачів. У нашій системі використовуємо метод косинусної схожості для визначення схожості між векторами тексту фільмів, щоб рекомендувати схожі фільми на основі контенту.

Косинусна схожість є метрикою, яка вимірює кут між двома векторами в багатовимірному просторі. Значення косинусної схожості варіюється від -1 до 1, де 1 означає, що вектори ідеально схожі, 0 - що вони ортогональні, а -1 - що вони повністю протилежні. Використання косинусної схожості дозволяє оцінити схожість між фільмами, не враховуючи їхню абсолютну величину, а тільки напрямок.

Для обчислення косинусної схожості між векторами використовуємо функцію `cosine_similarity` з бібліотеки `scikit-learn`. Реалізацію цих кроків в коді зображено на рис. 3.11:



```
from sklearn.metrics.pairwise import cosine_similarity
✓ 0.0s

similarity = cosine_similarity(vectors)
✓ 3.2s
```

Рисунок 3.11 – Застосування методу cosine_similarity

Функція рекомендації `recommend` приймає назву фільму як вхідний параметр і виводить п'ять найсхожіших фільмів.

Функція собою представляє знаходження індексу фільму в наборі даних за його назвою, обчислення косинусної схожості між обраним фільмом та усіма іншими фільмами, сортування списку фільмів за значенням схожості в порядку спадання та відбір перших п'яти фільмів (окрім самого фільму), виведення назв п'яти рекомендованих фільмів.

Оцінка рекомендацій `evaluate_recommendations` оцінює якість рекомендацій за допомогою метрик: середньоквадратична похибка (MSE), середня абсолютна похибка (MAE), точність, повнота та F1-міра.

Ці метрики використовуються для оцінки якості бінарних класифікацій. Спочатку вектори бінаризуються за допомогою порогу, що дорівнює середньому значенню істинного вектора. Потім обчислюються точність, повнота та F1-міра.

Функції рекомендації в нашій системі дозволяють користувачам отримувати релевантні пропозиції фільмів на основі схожого вмісту. Використання косинусної схожості забезпечує ефективне та точне визначення найбільш подібних фільмів, а функція оцінки рекомендацій дозволяє вимірювати якість рекомендацій і на їх основі приймати рішення щодо того який метод працює краще. Ці функції є важливими для забезпечення високої

якості обслуговування користувачів та підтримки їхнього інтересу до системи рекомендацій.

3.6 Аналіз результатів роботи реалізованого продукту

Результатом реалізованого продукту є виведення рекомендації користувачеві на основі обраного фільму у вигляді списку з 5 фільмів та оцінкою точності у вигляді коефіцієнту середньоквадратичної помилки та середньої абсолютної помилки. В цьому розділі розглянемо результати роботи програми для вибірки з декількох фільмів, порівняємо роботу для методів TF-IDF, CountVectorizer та методу машинного навчання KNN. Розглянемо метаінформацію для рекомендованих фільмів та фільмів на основі яких було зроблено рекомендацію, що дозволить зрозуміти природу градації самих оцінок, на основі яких робиться рішення щодо рекомендації.

Для аналізу роботи рекомендаційної системи обрали кілька популярних фільмів з різних жанрів та епох: Titanic, Cars, Moneyball, Inception. За розробленою програмою було отримано результати її роботи для обраних фільмів, як продемонстровано на рисунках наведених нижче.

Titanic, TF-IDF на рис. 3.12:

```
recommend('Titanic')
✓ 0.0s

Raise the Titanic
Ghost Ship
Poseidon
In the Heart of the Sea
Pirates of the Caribbean: On Stranger Tides

Mean Squared Error: 0.0002843955457007855
Mean Absolute Error: 0.0015248377944807585
```

Рисунок 3.12 – Результати роботи методу TF-IDF для фільму Titanic

Результатом роботи програмного продукту було отримано список фільмів:

Raise the Titanic

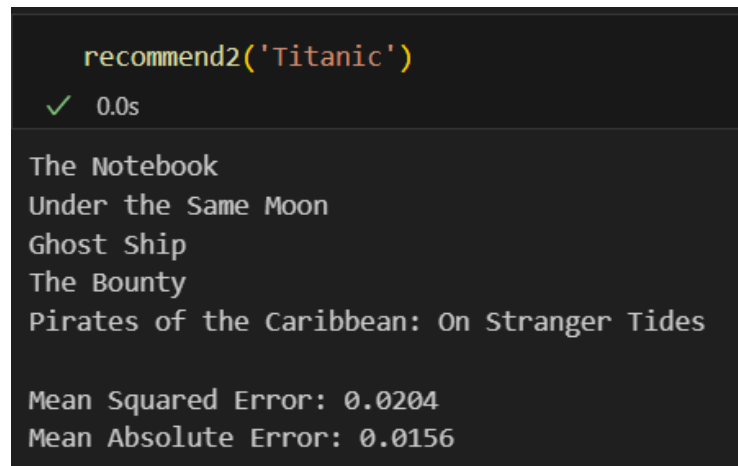
Ghost Ship

Poseidon

In the Heart of the Sea

Pirates of the Caribbean: On Stranger Tides

Titanic, CountVectorizer на рис. 3.13:



```

recommend2('Titanic')
✓ 0.0s
The Notebook
Under the Same Moon
Ghost Ship
The Bounty
Pirates of the Caribbean: On Stranger Tides

Mean Squared Error: 0.0204
Mean Absolute Error: 0.0156

```

Рисунок 3.13 – Результати роботи методу CountVectorizer для фільму Titanic

Результатом роботи програмного продукту було отримано список фільмів:

The Notebook

Under the Same Moon

Ghost Ship

The Bounty

Pirates of the Caribbean: On Stranger Tides

Titanic, KNN на рис. 3.14:

```
KNN:
The Boys from Brazil
Year One
Spaceballs
Sphinx
Patton

Mean Squared Error: 0.00039616872129564297
Mean Absolute Error: 0.0017597664291045991
```

Рисунок 3.14 – Результати роботи методу KNN для фільму Titanic

Результатом роботи програмного продукту було отримано список фільмів:

The Boys from Brazil
 Year One
 Spaceballs
 Sphinx
 Patton

Cars, TF-IDF на рис. 3.15:

```
recommend('Cars')
✓ 0.0s

Cars 2
The Fast and the Furious: Tokyo Drift
McFarland, USA
The Fast and the Furious
Little Miss Sunshine

Mean Squared Error: 0.0002539176126629287
Mean Absolute Error: 0.0015582642500985646
```

Рисунок 3.15 – Результати роботи методу TF-IDF для фільму Cars

Результатом роботи програмного продукту було отримано список фільмів:

Cars 2

The Fast and the Furious: Tokyo Drift

McFarland, USA

The Fast and the Furious

Little Miss Sunshine

Cars, CountVectorizer на рис. 3.16:

```
CountVectorizer:
Cars 2
Marmaduke
Rango
Up
The Croods

Mean Squared Error: 0.0098
Mean Absolute Error: 0.0094
```

Рисунок 3.16 – Результати роботи методу CountVectorizer для фільму
Cars

Результатом роботи програмного продукту було отримано список фільмів:

Cars 2

The Croods

Curious George

The Simpsons Movie

Marmaduke

Cars, KNN на рис. 3.17:

```

KNN:
An Everlasting Piece
3 Backyards
Death Race 2000
Teenage Mutant Ninja Turtles III
Watchmen

Mean Squared Error: 0.00039853658267207114
Mean Absolute Error: 0.002239539315124382

```

Рисунок 3.17 – Результати роботи методу KNN для фільму Cars

Результатом роботи програмного продукту було отримано список фільмів:

An Everlasting Piece
 3 Backyards
 Death Race 2000
 Teenage Mutant Ninja Turtles 3
 Watchmen

Moneyball, TF-IDF на рис. 3.18:

```

recommend('Moneyball')
✓ 0.0s

42
The Benchwarmers
A League of Their Own
Million Dollar Arm
Mr. 3000

Mean Squared Error: 0.00027449764166996007
Mean Absolute Error: 0.0013781904823954483

```

Рисунок 3.18 – Результати роботи методу TF-IDF для фільму Moneyball

Результатом роботи програмного продукту було отримано список фільмів:

42

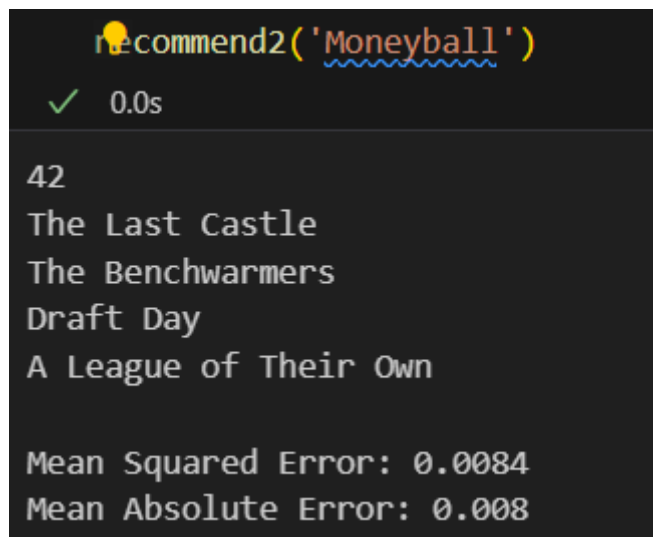
The Benchwarmers

A League of Their Own

Million Dollar Arm

Mr. 3000

Moneyball, CountVectorizer на рис. 3.19:



```

recommend2('Moneyball')
✓ 0.0s
42
The Last Castle
The Benchwarmers
Draft Day
A League of Their Own

Mean Squared Error: 0.0084
Mean Absolute Error: 0.008

```

Рисунок 3.19 – Результати роботи методу CountVectorizer для фільму Cars

Результатом роботи програмного продукту було отримано список фільмів:

42

The Last Castle

The Benchwarmers

Draft Day

A League of Their Own

Moneyball, KNN на рис. 3.20:

```
KNN:
Battle of the Year
The Timber
Sherrybaby
Superbad
Held Up

Mean Squared Error: 0.0003874470173460296
Mean Absolute Error: 0.0015987265440599305
```

Рисунок 3.20 – Результати роботи методу KNN для фільму Cars

Результатом роботи програмного продукту було отримано список фільмів:

Battle of the Year
The Timber
Sherrybaby
Superbad
Held Up

Inception, TF-IDF на рис. 3.21:

```
recommend('Inception')
✓ 0.2s

Don Jon
Premium Rush
Cypher
Hesher
Duplex

Mean Squared Error: 0.00033245666488278855
Mean Absolute Error: 0.0018795491033086265
```

Рисунок 3.21 – Результати роботи методу TF-IDF для фільму Inception

Результатом роботи програмного продукту було отримано список фільмів:

Don Jon

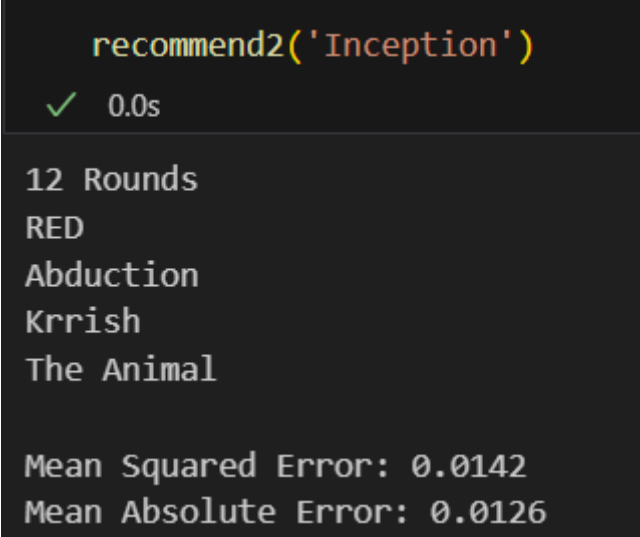
Premium Rush

Cypher

Hesher

Duplex

Inception, CountVectorizer на рис. 3.22:



```
recommend2('Inception')
✓ 0.0s
12 Rounds
RED
Abduction
Krrish
The Animal

Mean Squared Error: 0.0142
Mean Absolute Error: 0.0126
```

Рисунок 3.22 – Результати роботи методу CountVectorizer для фільму Inception

Результатом роботи програмного продукту було отримано список фільмів:

12 Rounds

RED

Abduction

Krrish

The Animal

Inception, KNN на рис. 3.23:

```
KNN:
Better Luck Tomorrow
Unnatural
Safety Not Guaranteed
Resident Evil: Apocalypse
Bullet to the Head

Mean Squared Error: 0.0004
Mean Absolute Error: 0.0017980064337212369
```

Рисунок 3.23 – Результати роботи методу KNN для фільму Inception

Результатом роботи програмного продукту було отримано список фільмів:

12 Rounds
RED
Abduction
Krrish
The Animal

Контекстуально можна побачити навіть з назв рекомендованих фільмів, що вони є подібними фільмами, а оцінки похибок є низькими, отже можна зробити висновки за сумою всіх ознак, що програма працює коректно і дійсно рекомендує фільми на основі обраного, які можуть зацікавити користувача.

Усі методи показують високий рівень схожості в рекомендаціях, з деякими варіаціями у виборі фільмів. Випадки, коли результати відрізняються, можуть бути пояснені різницею в підході до врахування частоти термінів. Зберемо значення оцінок точності для кожного фільму та методу в одній табл. 3.1 для зручності порівняння:

Таблиця 3.1 Порівняння метрик

	TF-IDF		CountVectorizer		KNN	
	MSE	MAE	MSE	MAE	MSE	MAE
Titanic	0.0002844	0.001525	0.0204	0.0156	0.000396	0.001759
Cars	0.00025392	0.00156	0.0108	0.01	0.000398	0.002239
Moneyball	0.0002745	0.0013781	0.0084	0.008	0.000398	0.001598
Inception	0.0003325	0.00188	0.0142	0.0126	0.0004	0.001798

З аналізу складеної таблиці можемо побачити, що метод KNN відпрацьовує краще, ніж CountVectorizer, але гірше ніж TF-IDF тому що значення оцінок якості там більш наближені до 0 (про оцінки було написано у Розділі 2).

Розглянемо на прикладі рекомендацій фільму Titanic метадані для фільмів які рекомендуються одним методом, але не рекомендуються іншим, щоб наочно зрозуміти різницю в векторах.

Метаінформація Titanic:

“84 years later, a 101-year-old woman named rose dewitt bukater tells the story to her granddaughter lizzy calvert, brock lovett, lewis bodine, bobby buell and anatoly mikailavich on the keldysh about her life set in april 10th 1912, on a ship called titanic when young rose boards the departing ship with the upper-class passengers and her mother, ruth dewitt bukater, and her fiancé, caledon hockley. meanwhile, a drifter and artist named jack dawson and his best friend fabrizio de rossi win third-class tickets to the ship in a game. and she explains the whole story from departure until the death of titanic on its first and last voyage april 15th, 1912 at 2:20 in the morning. drama romance thriller shipwreck iceberg ship panic titanic oceanliner epic richwoman-poorman love disaster tragiclove classdifferences imax starcrossedlovers steerage salvage richsnob 3d 1910s katewinslet leonardodicaprio francesfisher jamescameron”

Метаінформація Raise the Titanic:

“to obtain a supply of a rare mineral, a ship raising operation is conducted for the only known source, the titanic. action drama thriller m.emmetwalsh richardjordan davidselby jerryjameson”

Метаінформація Poseidon:

“a packed cruise ship traveling the atlantic is hit and overturned by a massive wave, compelling the passengers to begin a dramatic fight for their lives. adventure action drama thriller newyear'seve fire drowning cataclysm lossoffather atlanticoccean ball self-abandonment shipwreck giantwave blackout ship daughter single escape capsizedship kurtrussell richarddreyfuss joshlucas wolfgangpetersen”

Метаінформація The Notebook:

“an epic love story centered around an older man who reads aloud to a woman with alzheimer's. from a faded notebook, the old man's words bring to life the story about a couple who is separated by world war ii, and is then passionately reunited, seven years later, after they have taken different paths. romance drama poem river sadness fight loveofone'slife dementia class secretlove tears candle mailbox rachelmcadams ryangosling genarowlands nickcassavetes”

Метаінформація Under the Same Moon:

“tells the parallel stories of nine-year-old carlitos and his mother, rosario. in the hopes of providing a better life for her son, rosario works illegally in the u.s. while her mother cares for carlitos back in mexico. drama independentfilm womandirector katedelcastillo adriánalonso barona eugenioderbez patriciariggen”

TF-IDF – це метрика, яка використовується для визначення важливості термінів в документі в контексті колекції документів. Ось як вона працює на прикладі цих фільмів:

Titanic, ключові слова: drama, romance, shipwreck, iceberg, love, disaster, 1910s, katewinslet, leonardodicaprio, jamescameron

Raise the Titanic, ключові слова: action, drama, thriller, titanic, richardjordan, davidselby

Poseidon, ключові слова: adventure, action, drama, thriller, shipwreck, giantwave, kurtrussell, wolfgangpetersen

The Notebook, ключові слова: romance, drama, love, alzheimer's, rachelmcadams, ryangosling, genarowlands, nickcassavetes

Under the Same Moon, ключові слова: drama, independentfilm, katedelcastillo, adriáanalonsobarona, eugenioderbez, patriciariggen

TF-IDF визначає важливість кожного терміну у документі в контексті інших документів у колекції. Отже, TF-IDF буде приділяти увагу унікальним та важливим термінам для кожного фільму.

Порівняння між Titanic, Raise the Titanic, і Poseidon:

Titanic та Raise the Titanic:

Обидва фільми мають спільне ключове слово "Titanic".

"Raise the Titanic" має додаткові ключові слова "action", "thriller", "rare mineral", які є важливими для контексту фільму.

Titanic та Poseidon:

Обидва фільми мають спільні ключові слова "shipwreck", "drama", "thriller".

"Poseidon" має ключові слова "adventure", "action", "cruise ship", "massive wave", які відрізняють його від "Raise the Titanic".

Порівняння між Titanic, The Notebook, і Under the Same Moon:

Titanic та The Notebook:

Спільні ключові слова: "drama", "romance", "love".

"The Notebook" має ключові слова "alzheimer's", "older man", "reunited", які є унікальними в контексті цього фільму.

Titanic та Under the Same Moon:

Спільні ключові слова: "drama", "independentfilm".

"Under the Same Moon" має ключові слова "parallel stories", "illegal", "better life", "mexico", що є унікальними.

CountVectorizer просто підраховує частоту входження кожного слова у документі. Він не приділяє уваги на важливість слова у контексті всієї колекції.

CountVectorizer складає список усіх унікальних слів у всіх текстах та обчислює, скільки разів кожне слово з'являється у кожному тексті. Таким чином, фільми, які мають схожий розподіл слів, будуть між собою схожі. Розглянемо це для обраних прикладів у табл. 3.2 порівняння за CountVectorizer:

Таблиця 3.2 Порівняння за методом CountVectorizer

Фільм1	Фільм2	Ознаки(слова)
Titanic	The Notebook	love, drama, romance, tragedy, epic, 1910s
Titanic	Under the Same Moon	drama, parallel stories, illegal, better life, mexico
Raise the Titanic	Poseidon	action, drama, thriller, shipwreck, disaster

Також розглянемо для обраних прикладів у табл. 3.3 порівняння за методом TF-IDF:

Таблиця 3.3 Порівняння за методом TF-IDF

Фільм1	Фільм2	Ознаки та їх TF-IDF значення
Titanic	Raise the Titanic	drama (Titanic: 0.22, Raise the Titanic: 0.25), romance (Titanic: 0.19), shipwreck (Titanic: 0.21), iceberg (Titanic: 0.19), love (Titanic: 0.16), disaster (Titanic: 0.17), 1910s (Titanic: 0.16), action (Raise the Titanic: 0.25), thriller (Raise the Titanic: 0.25), rare mineral (Raise the Titanic: 0.25)

Продовження таблиці 3.3

Titanic	Poseidon	drama (Titanic: 0.22, Poseidon: 0.20), shipwreck (Titanic: 0.21, Poseidon: 0.19), thriller (Poseidon: 0.24), adventure (Poseidon: 0.20), action (Poseidon: 0.20), giantwave (Poseidon: 0.19)
Raise the Titanic	Poseidon	action (Raise the Titanic: 0.25, Poseidon: 0.20), thriller (Raise the Titanic: 0.25, Poseidon: 0.24), shipwreck (Poseidon: 0.19), drama (Raise the Titanic: 0.22, Poseidon: 0.20)

Ці таблиці демонструють, як CountVectorizer просто підраховує частоту слів у кожному тексті і рекомендує фільми, які мають схожий розподіл слів. TF-IDF, натомість, враховує важливість слова для конкретного документа у контексті усієї колекції документів, що робить його корисним для знаходження тематично спільних фільмів.

Метод К-найближчих сусідів (KNN) на основі векторизації метаінформації фільмів за допомогою TF-IDF дозволив отримати рекомендації, які містять спільні тематичні елементи з оригінальним фільмом "Titanic".

The Boys from Brazil має спільні ключові слова "drama", "thriller", "epic", "love", "disaster", що робить його тематично близьким до "Titanic".

Year One містить елементи "drama", "comedy", "adventure", "love", "historical", що також відображають деякі аспекти "Titanic".

Spaceballs включає "drama", "comedy", "adventure", "epic", "space", що надає зв'язок через епічні та пригодницькі елементи.

Sphinx ділиться ключовими словами "drama", "thriller", "adventure", "historical", "love", що відображають пригодницьку та драматичну природу "Titanic".

Patton має спільні елементи "drama", "historical", "epic", "war", "leadership", що підкреслюють історичну та епічну складову "Titanic".

Таким чином, метод KNN ефективно знаходить фільми з подібними тематичними елементами, що забезпечує релевантні рекомендації на основі вмісту, що наочно продемонстровано в табл. 3.4:

Таблиця 3.4 Порівняння результатів методу KNN

Фільм1	Фільм2	Ознаки(слова)
Titanic	The Boys from Brazil	drama, thriller, epic, love, disaster, historical, conflict, investigation, suspense, conspiracy, rescue, danger, secrets
Titanic	Patton	drama, historical, epic, leadership, strategy, conflict, biography, heroism, command, perseverance, courage, bravery
Patton	The Boys from Brazil	drama, historical, epic, conflict, strategy, investigation, conspiracy, biography, heroism, command, perseverance, courage, bravery

На основі оцінок та наочного порівняння можна зробити висновок, що метод TF-IDF працює краще для рекомендаційної системи на основі вмісту. Він забезпечує більш точні та тематично пов'язані рекомендації завдяки

врахуванню важливості ключових слів у контексті всієї колекції. Це дозволяє створювати більш релевантні рекомендації для користувачів, що підвищує їх задоволення від використання системи.

Отже, було доведено, що метод TF-IDF є ефективнішим для побудови рекомендаційної системи на основі вмісту у порівнянні з CountVectorizer та KNN, що було підтверджено на основі нашого аналізу та порівняння метаінформації фільмів.

3.7 Розробка інтерфейсу та приклад візуалізації

Для розробки інтерфейсу системи рекомендацій було використано кілька важливих бібліотек та інструментів. Основним інструментом для створення веб-інтерфейсу є бібліотека Streamlit, яка забезпечує зручний спосіб інтеграції інтерактивних елементів, таких як кнопки, випадаючі списки та інші віджети. Streamlit дозволяє швидко і легко створювати інтерактивні веб-додатки для обробки даних та машинного навчання, що значно спрощує процес розробки та впровадження системи.

Використовуючи інструменти Streamlit, такі як `st.set_page_config`, `st.title`, `st.selectbox`, `st.button`, `st.columns`, `st.write` та `st.image`, можемо налаштовувати конфігурацію сторінки, встановлювати заголовок, створювати випадаючі списки для вибору фільмів, додавати кнопки для отримання рекомендацій, розділяти сторінку на колонки для відображення інформації та відображати текстову інформацію і зображення.

Для виконання HTTP-запитів була використана бібліотека Requests, яка дозволяє отримувати інформацію про фільми та їх постери з API The Movie Database (TMDb). Ця бібліотека забезпечує простий і зручний спосіб взаємодії з зовнішніми API, що робить процес отримання даних швидким та ефективним.

Реалізований функціонал системи рекомендацій фільмів представляє собою пошукову строку, яка дозволяє користувачу отримати рекомендації на основі вибраного фільму. Інтерфейс системи надає користувачу можливість вибору фільму з випадającego списку, що містить назви всіх доступних фільмів. Після вибору фільму користувач може натиснути кнопку "Recommend", яка ініціює процес обробки запиту. Якщо такого фільму немає, то буде виведено напис: «No results» і програма не буде надавати можливість запросити рекомендацію.

Система обробляє запит шляхом обчислення подібності між обраним фільмом та іншими фільмами з використанням попередньо обробленої матриці подібності. Ця матриця містить значення подібності між різними фільмами, що дозволяє визначити, які фільми найбільше схожі на обраний користувачем фільм.

Після обробки запиту система надає список з п'яти рекомендованих фільмів. Кожен рекомендований фільм відображається у вигляді постера з відповідною назвою. Під постером фільмів є додаткова інформація про фільм, зокрема його опис та дату випуску. Це забезпечує користувачу зручний доступ до інформації про рекомендовані фільми та покращує загальний користувацький досвід.

Таким чином, реалізований інтерфейс забезпечує зручний та інтерактивний спосіб отримання рекомендацій фільмів. Користувач може швидко і легко обрати фільм з випадającego списку, отримати рекомендації на основі своїх вподобань та переглянути детальну інформацію про рекомендовані фільми. Це робить процес пошуку нових фільмів цікавим та зручним, надаючи користувачу всю необхідну інформацію в одному місці.

Приклад використання розробленого програмного інтерфейсу зображено на рис. 3.24-3.27:

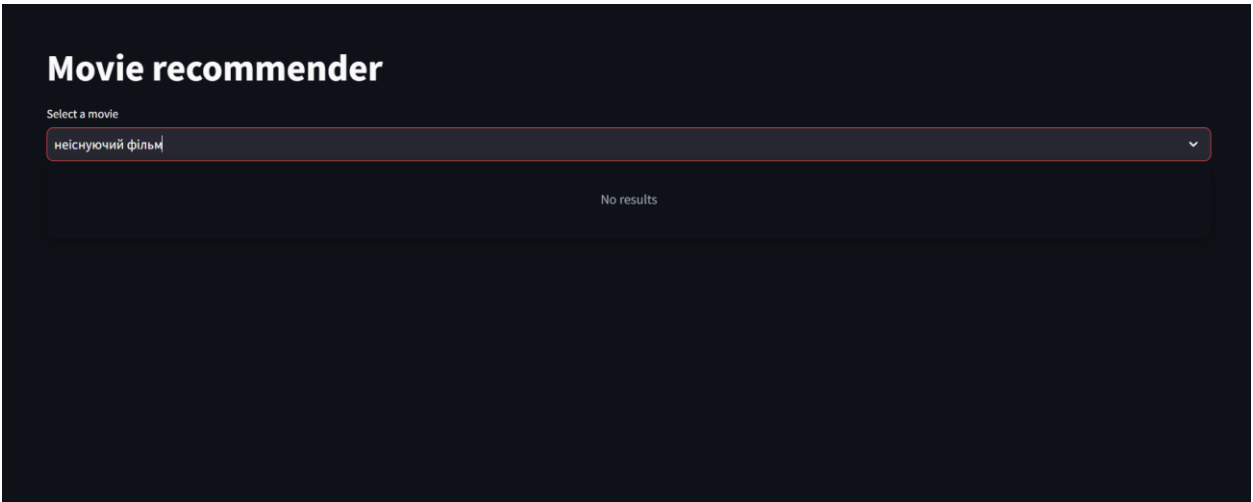


Рисунок 3.24 – Приклад використання пошуку для неіснуючого фільму

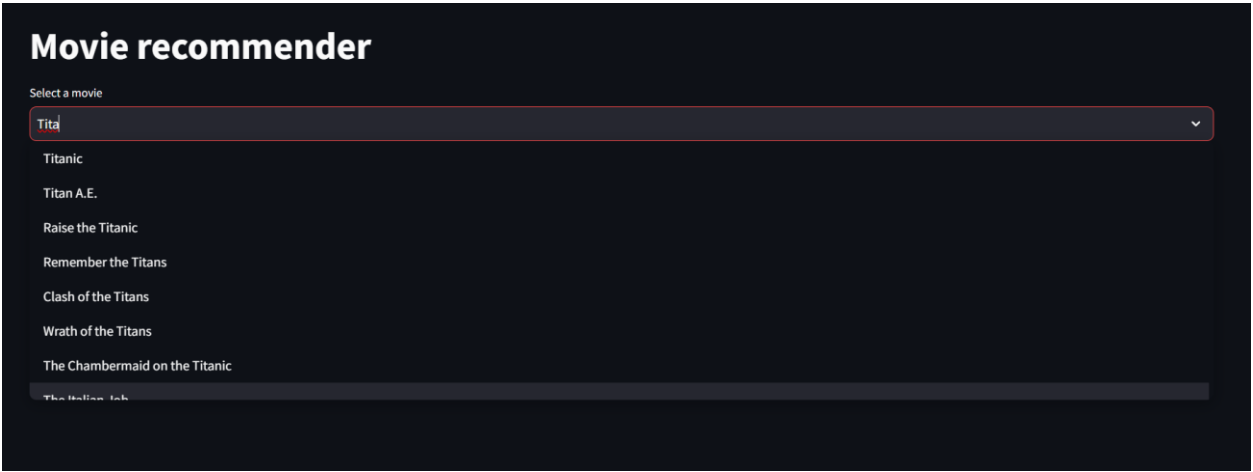


Рисунок 3.25 – Приклад використання пошуку для існуючого фільму

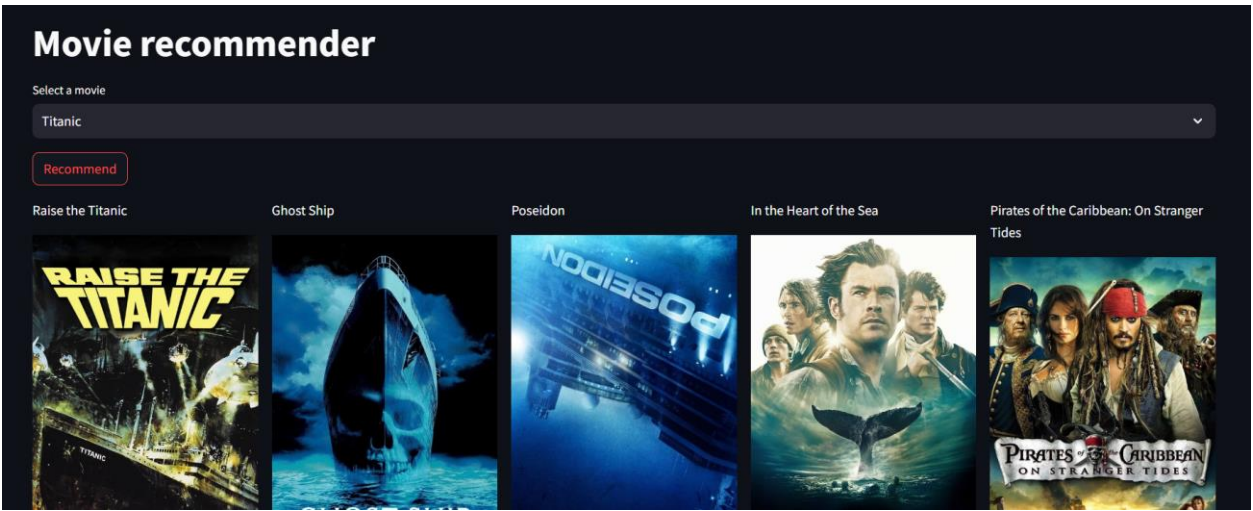


Рисунок 3.26 – Приклад роботи верхньої частини інтерфейсу

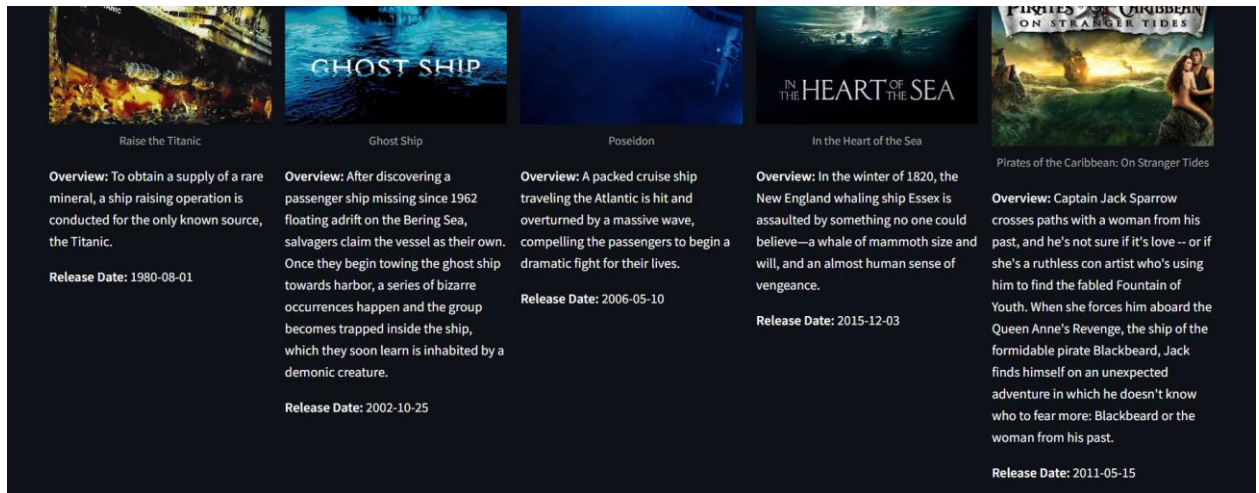


Рисунок 3.27 – Приклад роботи нижньої частини інтерфейсу

3.8 Висновки до розділу 3

Досліджено сучасні бібліотеки для обробки даних та машинного навчання, такі як pandas, numpy, nltk, sklearn та scipy. Кожна з цих бібліотек відіграє важливу роль у підготовці, аналізі та моделюванні даних, що дозволяє створювати потужні та ефективні рекомендаційні системи. Розроблено архітектуру програмного продукту рекомендаційної системи на основі content-based методу. Досліджено практичне застосування методів створення рекомендацій на основі вмісту та зроблено порівняння між методами TF-IDF, CountVectorizer та KNN, результатом якого було зроблено висновок, що метод TF-IDF є найточнішим серед усіх реалізованих. Створено графічний інтерфейс у вигляді веб-додатку для зручного користування програмним продуктом.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Формування завдання проектування

Даний розділ призначений для аналізу функціонально-вартісних аспектів програмного продукту, який був створений у цьому дослідженні. Це дозволить оцінити прийняті рішення щодо використаних технологій у контексті теорії вибору опцій та оптимізації виробництва. Ці елементи є критичними для подальшого застосування результатів цього проекту у реальних ситуаціях.

Вибір найбільш ефективної стратегії розробки будь-якої системи, особливо тієї, яка має практичне значення, є надзвичайно важливим кроком. Всі етапи розробки піддаються ризикам, особливо початкові, які визначають межі того, як дослідник або команда можуть просуватися до досягнення мети проекту. Неправильний збір та обробка даних, вибір середовища розробки, інструментів розробки, засобів моніторингу виконання етапів та належна презентація вже виконаної роботи можуть підірвати доцільність всього програмного продукту, його надійність і, що так само важливо, економічне обґрунтування.

Даний проект спрямований на розробку застосунку для рекомендації фільмів на основі аналізу вподобань користувачів, має науково-дослідницький характер та не передбачає комерційної реалізації. З цього приводу важливо зробити висновки згідно проведеного аналізу результатів, які можуть бути в подальшому використані фахівцями у галузі розробки рекомендаційних систем та аналітики даних.

Все ж таки, економічний аспект лишається важливим у плануванні, оскільки переважна частина проектів обмежена в часі, а одним з найпотужніших інструментів впливу на нього виступають гроші. Також варто

зазначити майбутню перспективу, яка полягає в тому, що науково-дослідницькі проекти, спрямовані на пошук новаторських підходів, доволі часто отримують належне фінансування лише за видимих результатів, що ще більше підкреслює значення вартісного аналізу проведеної роботи. Таким чином, буде зрозуміло, наскільки обґрунтовано проводити розробку подібних методів на основі вибраних технологій та отриманих результатів, які можуть бути проаналізовані при функціональному аналізі.

Слід зазначити, що аналіз ринку та конкурентів не можливий у традиційному розумінні. Це пояснюється тим, що творці передових рішень у галузі розробки рекомендаційних систем для фільмів не розкривають фінансових аспектів, а їх програмні продукти не розповсюджуються у вільному доступі з різних причин. Тому рішення не призначені для продажу і не мають на меті пряме впровадження у комерційні системи. Вони, скоріше, спрямовані на поширення знань та розуміння процесів у рекомендаційних системах та впливу зовнішніх і внутрішніх факторів на їх ефективність.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який дозволяє користувачеві взаємодіяти з базою даних наявних фільмів, обираючи з наявних фільмів, до якого потрібно створити рекомендацію подібних. Беручи за основу дану функцію, було виділено наступні головні функції виробу:

F_1 – вибір мови програмування для опрацювання даних.

F_2 – вибір бібліотеки створення інтерфейсу.

F_3 – вибір методу використання програмою пам'яті програми.

Зокрема, варто визначити декілька можливих варіантів їх реалізації:

Функція F_1 :

А) Мова програмування Python;

Б) Мова програмування C++;

Функція F_2 :

- А) PyQt5
- Б) Streamlit

Функція F_3 :

- А) Мануальне розподілення пам'яті
- Б) Автоматичне розподілення пам'яті

Наступним кроком було створення морфологічної карту, використовуючи описані вище варіанти реалізації основних функції. Кінцевий результат зображений на рис. 4.1.

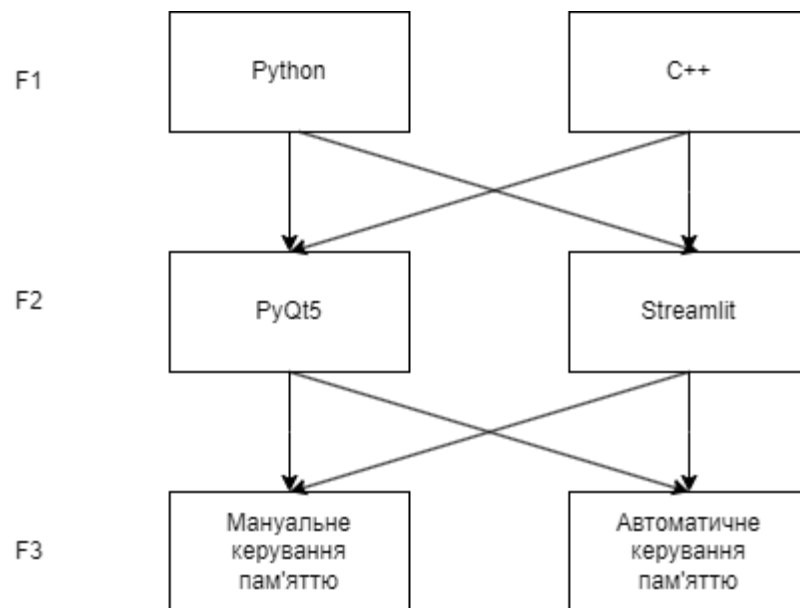


Рисунок 4.1 – Морфологічна карта

На основі визначених функцій, представлених варіантів реалізації та сформованої морфологічної карти, було побудовано позитивно-негативну матрицю. Результати якої продемонстровано нижче, у табл. 4.1.

Таблиця 4.1 Позитивно-негативна матриця

Функції	Варіанти	Переваги	Недоліки
F_1	А	Менший об'єм програмного коду, наявність бібліотек розроблених для обробки даних, оптимізоване написання коду	Не дуже ефективна для виконання великих обчислень або обробки великих обсягів даних
	Б	Швидке виконання розробленого програмного продукту	Не дуже ефективна для розробки програмного продукту, мала кількість бібліотек розроблених для обробки даних
F_2	А	Потужна та гнучка, широкі можливості для створення інтерфейсів	Може бути складним для початківців, може вимагати більше часу на розробку
	Б	Простий у використанні, вбудований у стандартну бібліотеку Python	Обмежений можливостями порівняно з іншими бібліотеками для створення інтерфейсів
F_3	А	Можливість тонкої оптимізації, контроль над розподілом пам'яті	Висока ймовірність помилок, складність в розробці та відладці
	Б	Простота використання, зменшення кількості помилок пов'язаних з пам'яттю	Непередбачуваність моментів очищення пам'яті, можливе збільшення витрат оперативної пам'яті

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 :

Оскільки час та обсяг написання програмного коду, наявність спеціалізованих бібліотек з машинного навчання для обробки великої кількості даних є більш вагомим фактором, ніж лише швидкість виконання, то варіант б) можна не розглядати.

Функція F_2 :

Так, як наведені варіанти реалізації мають свої значні переваги один над одним, варто розглянути та проаналізувати обидва.

Функція F_3 :

Оскільки використовуємо відносно невеликі датасети, то ймовірність виникнення несправностей у варіанті б) дуже низька, то варіант а) можна не розглядати через складність реалізації.

Отже, будемо розглядати такі варіанти реалізації ПП:

- $F_1\text{а} - F_2\text{а} - F_3\text{б}$
- $F_1\text{а} - F_2\text{б} - F_3\text{б}$

4.3 Обґрунтування системи параметрів програмного продукту

Даний підрозділ передбачає встановлення характеристик для попередньо аналізованих функцій програмного продукту. Таким чином, для розрахунку коефіцієнта технічного рівня, можемо розпочати з опису нашого проекту за допомогою наступних параметрів:

X_1 – об'єм оперативної пам'яті для проведення обчислень

X_2 – час обробки даних

X_3 – точність розв'язку

X_4 – об'єм програмного коду

Відповідно до вимог замовника та умов, що характеризують роботу програмного продукту, було обрано найгірші, середні та найкращі значення параметрів, як показано в таблиці 4.2.

Таблиця 4.2 Основні параметри програмного продукту

Параметр	Умовне позначення	Одиниця виміру	Значення		
			Найгірше	Середнє	Найкраще
Час обробки даних	X_2	Мс	750	450	60
Точність розв'язку	X_3	Доля одиниці	10E-1	10E-3	10E-3
Об'єм програмного коду	X_4	Кількість рядків коду	1800	1400	1000
Об'єм оперативної пам'яті для проведення обчислень	X_1	Мб	32	16	8

За отриманою таблицею основних параметрів програмного продукту було створено відповідні графічні представлення, які продемонстровано на рис. 4.2 - 4.5.

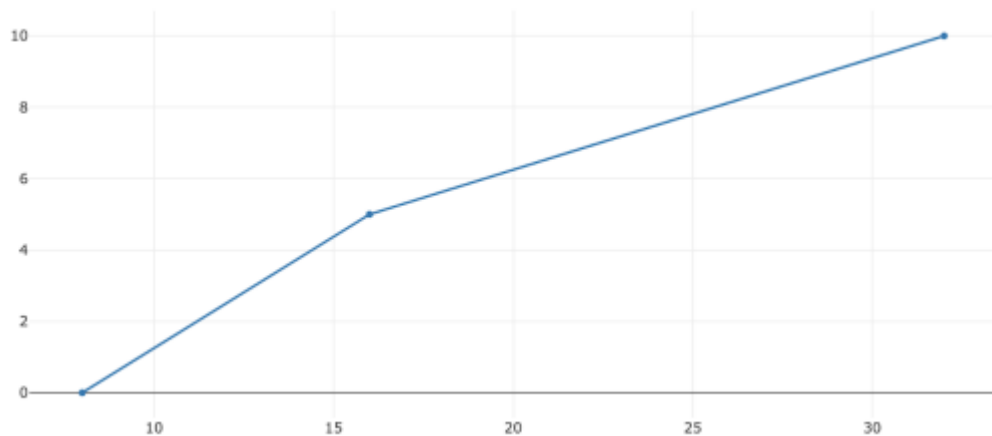


Рисунок 4.2 – Об'єм оперативної пам'яті для проведення обчислень

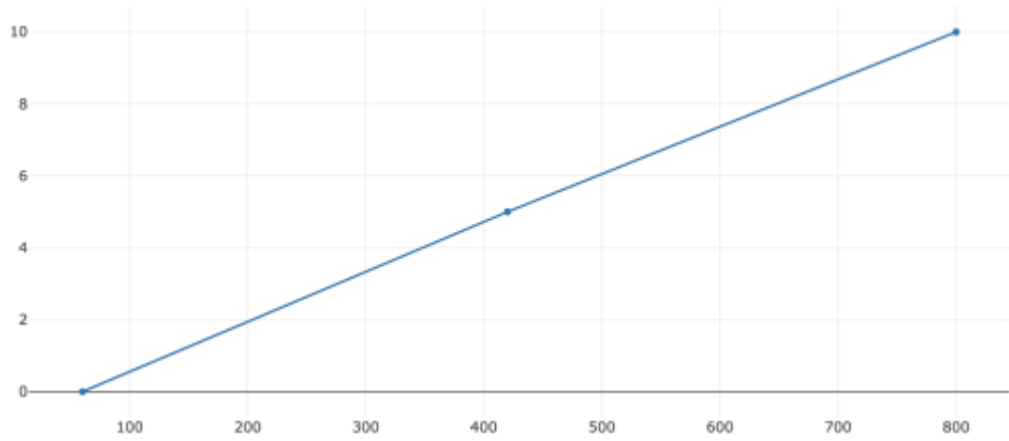


Рисунок 4.3 – Час обработки данных

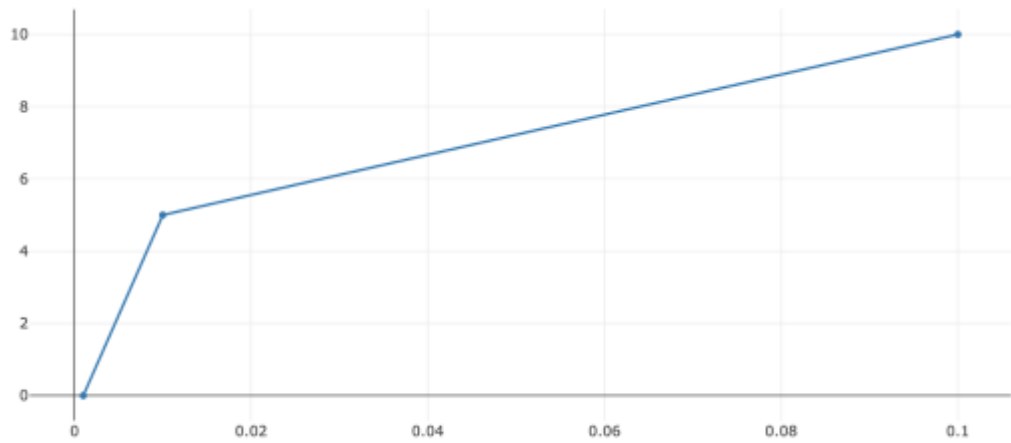


Рисунок 4.4 – Точность разв'язку

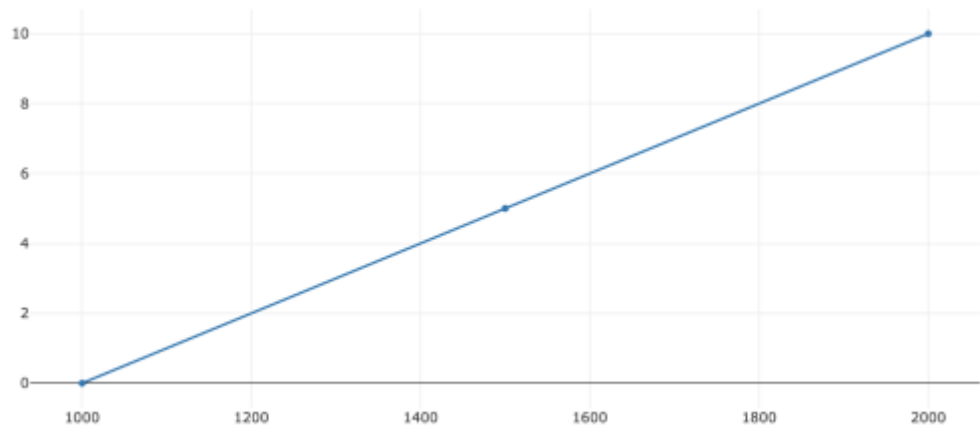


Рисунок 4.5 – Об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Четвертий етап функціонально-вартісного аналізу програмного продукту включає в себе визначення вагомості кожного параметру в загальній кількості розглядуваних під час оцінювання параметрів за допомогою методу попарного порівняння.

Для цього спочатку було знайдено коефіцієнти вагомості, визначено ступінь важливості параметрів, шляхом присвоєння їм різних рангів експертами-фахівцями в галузі. Результати продемонстровано в табл. 4.3.

Таблиця 4.3 Результати ранжування параметрів

Позначення параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
		1	2	3	4	5	6	7			
X_1	Мб	3	4	2	3	4	3	4	21	5.5	30.25
X_2	Кількість рядків	1	2	1	1	2	2	2	12	-6.5	42.25
X_3	Відсотки	2	1	3	2	1	1	1	11	-6.5	42.25
X_4	Секунди	4	3	4	4	3	4	3	26	7.5	56.25
Разом		10	10	10	10	10	10	10	70	0	171

Після чого було обчислено параметри:

1. Загальна сума рангів (4.1):

(4.1)

$$R_i = \sum_{j=1}^N r_{ij} = \frac{Nn(n+1)}{2} = \frac{7 \cdot 4(4+1)}{2} = 70$$

2. Середня сума рангів (4.2):

(4.2)

$$T = \frac{1}{n} R_i = 17.5$$

Продовження таблиці 4.4

X_2 і X_4	<	<	<	>	>	<	>	<	0.5
X_3 і X_4	>	>	>	>	>	>	>	>	0.5

Далі за результатами числових значень коефіцієнтів переваг на основі підсумкових оцінок, було складено квадратну матрицю $A = \|x_{ij}\|$ та розраховано пріоритетність K_{Bi} для кожного із параметрів допоки різниця між ітераціями розрахунку даних вагомості не стала меншими за 5%.

Розрахунки для першої ітерації відбувались за формулами 4.6 і 4.7:

(4.6)

$$K_{Bi} = \frac{b_i}{\sum_{i=1}^n b_i}$$

(4.7)

$$b_i = \sum_{i=1}^N a_{ij}$$

Для поточних ітерацій використовувались 4.8 і 4.9:

(4.8)

$$K_{Bi} = \frac{b_i'}{\sum_{i=1}^n b_i'}$$

(4.9)

$$b_i' = \sum_{i=1}^N a_{ij} b_j$$

Результати наведено в табл. 4.5:

Таблиця 4.5 Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Ітерація №1		Ітерація №2	
	X_1	X_2	X_3	X_4	b_i	K_{Bi}	b_i^1	K_{Bi}^1
X_1	1	1.5	1.5	0.5	4.5	0.30	16.25	0.29
X_2	0.5	1	1.5	0.5	3.5	0.20	12.25	0.19
X_3	0.5	0.5	1	0.5	2.5	0.25	9.25	0.25
X_4	1.5	1.5	1.5	1	5.5	0.25	21.25	0.25
Всього:					16	1	59	1

Оскільки різниця між коефіцієнтами вагомості параметрів вже на другій ітерації почала дорівнювати 2% (0.02), що вже є меншим від 5% (0.05), то було призупинено подальші розрахунки і визначено дану таблицю як остаточну.

4.5 Аналіз рівня якості варіантів реалізації функцій

На основі табл. 4.5, було оцінено потенційні абсолютні значення параметрів для нашого проекту. Кінцевим результатом оцінки виступає табл. 4.6, де проведено розрахунки показників технічного рівня, що враховує варіанти реалізації функцій та параметри на основі раніше обчисленого коефіцієнта.

Таблиця 4.6 Розрахунок показників рівня якості

Основні функції	Варіант реалізації функції	Параметри задіяні у реалізації	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	X_4	1100	1	0.344	0.344
F_2	А	X_1	12	4	0.281	1.124
		X_2	420	5	0.189	0.945
	Б	X_1	15	5	0.281	1.405
		X_2	350	4	0.189	0.756
F_3	Б	X_3	10E-2	5	0.156	0.78

Для обох варіантів реалізації системи (F_2 А та F_2 Б) було просумовано показники технічного рівня, отримавши наступні узагальнені коефіцієнти:

$$K_{F2A} = 1.124 + 0.945 + 0.78 + 0.344 = 3.193$$

$$K_{F2B} = 1.405 + 0.756 + 0.78 + 0.344 = 3.285$$

Таким чином, при порівнянні обох варіантів реалізації програмного продукту, функціональний надав можливість визначити найкращий варіант, яким виступив K_{F2B} , оскільки його значення більше на 0.92.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення загальної трудомісткості системи даного проекту було використано формулу 4.10:

(4.10)

$$T_O = T_P \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТМ}$$

Після чого визначено початкові дані системи завдання (розробка програмного продукту) з метою отримання даних про поправочні коефіцієнти для розрахунку загальної трудомісткості. Даний перелік включає в себе:

- Розробка проекту програмного продукту;
- Розробка користувацького інтерфейсу для демонстрації;

З даного переліку було сформовано вже поправочні коефіцієнти за таблицями умов із наданих методичних вказівок:

- $T_P - 90$
- $K_{\Pi} - 1.7$
- $K_{СК} - 1$
- $K_M - 1$
- $K_{СТ} - 0.7$
- $K_{СТМ} - 1$

Таким чином в наявності опинились всі елементи необхідні для складання початкового рівняння (4.11):

(4.11)

$$T_0 = 90 \cdot 1.7 \cdot 1 \cdot 1 \cdot 0.7 \cdot 1 \approx 107.1 \text{ (людино днів)}$$

Розрахувавши загальну трудомісткість програмного продукту, її також було розраховано для двох варіантів реалізації (4.12):

(4.12)

$$T_1 = (107.1 + 17.28 + 4.8 + 17.28) \cdot 8 \approx 1171.68 \text{ (людино днів)}$$

$$T_2 = (107.1 + 17.28 + 6.96 + 17.28) \cdot 8 \approx 1188.56 \text{ (людино днів)}$$

Результат розрахунків виявився доволі логічним, оскільки хоч підхід та функціонал в обох бібліотеках до створенню графічного інтерфейсу може відрізнятись, сама його структура з елементами повинна лишатись однаковою.

Оскільки у розробці бере участь один науковець у сфері даних з окладом 6000 грн та науковець з окладом 12000, то середня заробітна плата складає (4.13):

(4.13)

$$C_q = \frac{M}{T_m \cdot t} = \frac{6000 + 6000 + 12000}{3 \cdot 21 \cdot 8} = 47.61 \text{ (грн)}$$

Також було обчислено зарплату для обох варіантів реалізації (4.14):

(4.14)

$$\text{I. } C_{\text{зп}} = C_q \cdot T_i \cdot K_d = 47.61 \cdot 1171.68 \cdot 1.2 = 66953.14 \text{ (грн)}$$

$$\text{II. } C_{\text{зп}} = C_q \cdot T_i \cdot K_d = 47.61 \cdot 1188.56 \cdot 1.2 = 67904.80 \text{ (грн)}$$

Відраховано соціальний внесок від заробітної плати (4.15):

(4.15)

$$\text{I. } C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 66953.14 \cdot 0.22 = 14729 \text{ (грн)}$$

$$\text{II. } C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 67904.80 \cdot 0.22 = 14939 \text{ (грн)}$$

Для знаходження вартості однієї машино-години, спочатку було розраховано перелік інших змінних.

Машинний час (4.16), враховуючи додаткову заробітну плату і відрахування соціального внеску:

(4.16)

$$C_{\text{Г}} = 12 \cdot M \cdot K_3 = 12 \cdot 6000 \cdot 0.2 = 14400 \text{ (грн)}$$

$$C_{\text{зп}} = C_{\text{Г}} \cdot (1 + K_3) = 14400 \cdot 1.2 = 17280 \text{ (грн)}$$

$$C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 17280 \cdot 0.22 = 3807.6 \text{ (грн)}$$

Амортизаційні відрахування при вартості обладнання в 20 000 грн та амортизації в 25% (4.17):

(4.17)

$$C_{\text{А}} = K_{\text{ТМ}} \cdot K_{\text{А}} \cdot Ц_{\text{ПР}} = 1.15 \cdot 0.25 \cdot 20\,000 = 5750 \text{ (грн)}$$

Вартість ремонту та профілактики (4.18):

(4.18)

$$C_{\text{Р}} = K_{\text{ТМ}} \cdot K_{\text{Р}} \cdot Ц_{\text{ПР}} = 1.15 \cdot 0.05 \cdot 20\,000 = 1150 \text{ (грн)}$$

Ефективний годинний фонд часу ПК за рік (4.19):

(4.19)

$$\begin{aligned} T_{\text{ЕФ}} &= (D_{\text{К}} - D_{\text{В}} - D_{\text{С}} - D_{\text{Р}}) \cdot t_3 \cdot K_{\text{В}} = \\ &= (365 - 104 - 12 - 16) \cdot 8 \cdot 0.9 = 1706.4 \text{ (год.)} \end{aligned}$$

Кількість коштів витрачених на оплату електроенергії (4.20):

(4.20)

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_{\text{З}} \cdot C_{\text{ен}} \cdot K_{\text{В}} = 1706.4 \cdot 0.15 \cdot 0.8 \cdot 5.23 = 1085.9 (\text{грн.})$$

Накладні витрати (4.21):

(4.21)

$$C_{\text{Н}} = C_{\text{ПР}} \cdot 0.67 = 20\,000 \cdot 0.67 = 13\,400 (\text{грн.})$$

Вартість експлуатації (4.22):

(4.22)

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}} = 17\,280 + \\ + 3\,801.6 + 5\,750 + 1\,150 + 1\,085.9 + 13\,400 = 42467.5 (\text{грн.})$$

Вартість однієї машино-години (4.23):

(4.23)

$$C_{\text{М-Г}} = \frac{C_{\text{ЕКС}}}{T_{\text{ЕФ}}} = \frac{42467.5}{1706.4} = 24.89 (\text{грн.})$$

Для обох варіантів реалізації (4.24):

(4.24)

$$\text{I.} \quad C_{\text{М}} = C_{\text{М-Г}} \cdot T = 24.89 \cdot 1171.68 = 29\,163.15 (\text{грн.})$$

$$\text{II.} \quad C_{\text{М}} = C_{\text{М-Г}} \cdot T = 24.89 \cdot 1188.56 = 29\,583.26 (\text{грн.})$$

Накладні витрати (4.25):

(4.25)

- I. $C_H = C_{3П} \cdot 0.67 = 66\,953.14 \cdot 0.67 = 44\,858.60$ (грн.)
- II. $C_H = C_{3П} \cdot 0.67 = 67\,904.80 \cdot 0.67 = 45\,396.22$ (грн.)

Загальна вартість розробки програмного продукту за обома варіантами становила однакове значення, оскільки обидві варіанти не потребують особливих витрат на якісь підписки чи додаткові типи послуг (4.26):

(4.26)

$$C_{ПП} = C_{3П} + C_{ВІД} + C_M + C_H = 66\,953.14 + 24\,618 + 29\,163.15 + 44\,858.60 = 165\,592.89 \text{ (грн.)}$$

$$C_{ПП} = 66\,953.14 + 24\,618 + 29\,583.26 + 44\,858.60 = 166\,013 \text{ (грн.)}$$

4.7 Вибір кращого варіанта техніко-економічного рівня

Останнім етапом було обрано найкращий варіант реалізації. Для цього спочатку розраховано коефіцієнт техніко-економічного рівня для двох варіантів реалізації відповідно (4.27):

(4.27)

$$K_{\text{ТЕР } j} = \frac{K_{Kj}}{C_{ППj}} = \frac{3.193}{165\,592.89} = 19.28 \cdot 10^{-6}$$

$$K_{\text{ТЕР } j} = \frac{K_{Kj}}{C_{ППj}} = \frac{3.285}{166\,013} = 19.79 \cdot 10^{-6}$$

Отже, оскільки величина коефіцієнта техніко-економічного рівня визначає його ефективність, то було прийнято рішення імплементувати другий варіант, який налічує в собі наступні опції:

- Python;

- Streamlit;
- Автоматичне розподілення пам'яті.

4.8 Висновки до розділу 4

В результаті проведеного функціонально-вартісного аналізу програмного продукту, що було розроблено в рамках цієї роботи, отримано висновки стосовно ефективності обраних технологій та стратегій економічних рішень для даного продукту. Оцінки основних характеристик програмного забезпечення, спрямованого на рекомендацію фільмів на основі аналізу вподобань користувачів, виявили різноманітні варіанти реалізації, які забезпечують оптимальну стратегію вибору. Метод функціонально-вартісного аналізу було використано для об'єктивної оцінки реальної вартості продукту та виявлення можливостей зниження витрат. Це сприятиме не тільки подальшому потенційному впровадженню продукту в реальні умови, але й розвитку спеціаліста з точки зору архітектора життєвого шляху програмного продукту.

ВИСНОВКИ

Рекомендаційні системи є важливим елементом програмного продукту кожного сучасного онлайн бізнесу. Краща якість рекомендації, ніж у конкурента може стати визначальним фактором у боротьбі за лідерство на ринку між великими корпораціями та збільшити прибутковість продукту.

В ході роботи було досліджено сучасні методи генерації рекомендацій та визначено, які з них використовуються у рекомендаційних системах лідерів онлайн-бізнесу. Можна простежити тенденцію, що більшість компаній використовують колаборативні системи, використовуючи дані, які вони зібрали від користувачів. Проте не всі бізнес моделі здатні використовувати цю модель рекомендації, тож моделі на основі вмісту також залишаються популярними серед лідерів ринку.

В практичній частині роботи було розроблено систему, яка впроваджує такі методи, як TF-IDF та CountVectorizer на основі косинусною схожістю та KNN.

В ході проведених експериментів за допомогою таких метрик як MSE, MAE було виявлено, що метод TF-IDF з косинусною схожістю показує найкращі результати, метод KNN посів друге місце, а метод CountVectorizer продемонстрував найгірші результати.

Таким чином, було розроблено рекомендаційну систему відеострімінгово сервісу на основі вмісту, яка дозволяє отримати рекомендації фільмів шляхом застосування різних алгоритмів обчислення подібності між обраним фільмом та іншими фільмами використовуючи їх метадані. Для використання розробленої системи було також розроблено інтерфейс, який містить пошукову строку, що дозволяє користувачу ввести пошуковий запит та отримати рекомендації на основі вибраного фільму. Інтерфейс системи надає користувачу можливість вибору фільму з випадваючого списку, який

містить назви всіх доступних фільмів, що робить пошук зручним та інтуїтивно зрозумілим.

Варіантом продовження дослідження може бути розробка моделі під роботу з базою даних, що оновлюється автоматично з певним інтервалом часу та моделі для загальної оптимізації оцінок якості рекомендацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Guide to TF-IDF. URL: <https://builtin.com/articles/tfidf#:~:text=TF-IDF%20is%20a%20natural,machine%20learning%20model%20read%20words>
2. Balabanovic M., Shoham Y. Content-based, Collaborative Recommendation. Communications of the ACM, 1997, 40, P. 66–72. URL: <https://dl.acm.org/doi/10.1145/245108.245124>
3. Bernard J. Python Data Analysis with pandas. Python Recipes Handbook, 2019. URL: https://doi.org/10.1007/978-1-4842-0241-8_5
4. Collaborative based Filtering in Recommender Systems & Content Based Recommender Systems. URL: <https://medium.com/@chandu.bathula16/collaborative-based-filtering-in-recommender-systems-content-based-recommender-systems-9bb1bbd62682>
5. CountVectorizer vs TfidfVectorizer. URL: <https://medium.com/@shandeep92/countvectorizer-vs-tfidfvectorizer-cf62d0a54fa4>
6. Inside recommendations: how a recommender system recommends. URL: <https://medium.com/sciforce/inside-recommendations-how-a-recommender-system-recommends-9afc0458bd8f>
7. Jayalakshmi S., Ganesh N., Cep R. Movie Recommender Systems: Concepts, Methods, Challenges and Future Directions, 2022, 22, P. 5-8. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9269752/#sec4-sensors-22-04904>
8. Kenthapadi K., Le B., Venkataraman G. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. Eleventh ACM Conference on Recommender Systems, 2017, P. 346-347. URL: https://www.researchgate.net/publication/319285276_Personalized_Job_Recommendation_System_at_LinkedIn_Practical_Challenges_and_Lessons_Learned

9. K-Nearest Neighbors (KNN) Algorithm for Machine Learning. URL: <https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26>
10. Margi K. Comparison of the TF-IDF method with the count vectorizer to classify hate speech. Engineering, Mathematics and Computer Science Journal, 2023, 5, P. 79-83. URL: <https://journal.binus.ac.id/index.php/EMACS/article/view/9978>
11. Mohamed M. H., Khafagy M. H., Ibrahim M. H. Recommender Systems Challenges and Solutions Survey. 2019 International Conference on Innovative Trends in Computer Engineering (ITCE), Aswan, Egypt, 2019, P. 149-155. URL: https://www.researchgate.net/publication/331063850_Recommender_Systems_Challenges_and_Solutions_Survey
12. Nayek J., Das R. Evaluation of Famous Recommender Systems: A Comparative Analysis. Library Philosophy and practice, 2021, 5, P. 33-36. URL: https://www.researchgate.net/publication/351282198_Evaluation_of_Famous_Recommender_Systems_A_Comparative_Analysis
13. Netflix Movies and Series Recommendation Systems. URL: <https://pyimagesearch.com/2023/07/03/netflix-movies-and-series-recommendation-systems/>
14. Porter Stemming Algorithm – Basic Intro. URL: <https://vijinimallawaarachchi.com/2017/05/09/porter-stemming-algorithm/>
15. Prioritizing Home Attributes Based on Guest Interest. URL: <https://medium.com/airbnb-engineering/prioritizing-home-attributes-based-on-guest-interest-3c49b827e51a>
16. Radhika G. Evaluation of rule-based, CountVectorizer, and Word2Vec machine learning models for tweet analysis to improve disaster relief. IEEE Global Humanitarian Technology Conference, 2021, P. 16-19. URL: <https://ieeexplore.ieee.org/document/9612486>

17. Recommendation Algorithms And System Designs of Dropbox, Spotify, Airbnb, YouTube, Netflix And Uber. URL: <https://dev.to/johndoeofficial/recommendation-algorithms-and-system-designs-of-dropbox-spotify-airbnb-youtube-netflix-and-uber-10bf>
18. Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 2002, 34, P. 1–47. URL: <https://dl.acm.org/doi/10.1145/505282.505283>
19. TMDB Movie Dataset. URL: <https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata>
20. Vilakone P., Park D., Xinchang K. An Efficient movie recommendation algorithm based on improved k -clique. Hum. Cent. Comput. Inf. Sci, 2018, 8, P. 38. URL: <https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0161-6>
21. What Is a Movie Recommendation System in ML? URL: <https://labeleyourdata.com/articles/movie-recommendation-with-machine-learning>
22. What is Scikit-Learn? URL: <https://www.codecademy.com/article/scikit-learn>

ДОДАТОК А

```

import numpy as np
import pandas as pd
import ast
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import mean_absolute_error, mean_squared_error,
precision_score, recall_score, f1_score
from scipy.stats import pearsonr
import nltk

movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')

movies = movies.merge(credits, on='title')
movies = movies[['movie_id', 'title', 'overview', 'genres', 'keywords', 'cast',
'crew']]
movies.dropna(inplace=True)

def convert(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L

movies['genres'] = movies['genres'].apply(convert)
movies['keywords'] = movies['keywords'].apply(convert)

def convert3(obj):
    L = []
    counter = 0
    for i in ast.literal_eval(obj):
        if counter != 3:
            L.append(i['name'])
            counter += 1
        else:
            break
    return L

```

```

movies['cast'] = movies['cast'].apply(convert3)

def fetch_director(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L

movies['crew'] = movies['crew'].apply(fetch_director)
movies['overview'] = movies['overview'].apply(lambda x: x.split())

movies['genres'] = movies['genres'].apply(lambda x: [i.replace(" ", "") for i
in x])
movies['keywords'] = movies['keywords'].apply(lambda x: [i.replace(" ", "") for
i in x])
movies['cast'] = movies['cast'].apply(lambda x: [i.replace(" ", "") for i in
x])
movies['crew'] = movies['crew'].apply(lambda x: [i.replace(" ", "") for i in
x])

movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] +
movies['cast'] + movies['crew']
new_df = movies[['movie_id', 'title', 'tags']]

new_df['tags'] = new_df['tags'].apply(lambda x: " ".join(x))
new_df['tags'] = new_df['tags'].apply(lambda x: x.lower())

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

def stem(text):
    y = []
    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)

# TF-IDF
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
vectors = tfidf.fit_transform(new_df['tags']).toarray()
from sklearn.metrics.pairwise import cosine_similarity

```

```

# Cosine Similarity
similarity = cosine_similarity(vectors)

# CountVectorizer
cv = CountVectorizer(max_features=5000, stop_words='english')
vectors2 = cv.fit_transform(new_df['tags']).toarray()
similarity2 = cosine_similarity(vectors2)

movies['combined'] = movies['genres'].apply(lambda x: ' '.join(x)) + ' ' + \
    movies['keywords'].apply(lambda x: ' '.join(x)) + ' ' + \
    movies['cast'].apply(lambda x: ' '.join(x)) + ' ' + \
    movies['crew'].apply(lambda x: ' '.join(x))

knn_tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
knn_vectors = knn_tfidf.fit_transform(movies['combined']).toarray()

# KNN Model
model_knn = NearestNeighbors(metric='euclidean', algorithm='brute')
model_knn.fit(knn_vectors)

# Function for evaluating recommendations
def evaluate_recommendations(movie, recommended_movies, vectors):
    movie_index = new_df[new_df['title'] == movie].index[0]
    true_movie_vector = vectors[movie_index]
    predicted_movie_vector = vectors[recommended_movies[0]]
    mse = mean_squared_error(true_movie_vector, predicted_movie_vector)
    mae = mean_absolute_error(true_movie_vector, predicted_movie_vector)
    pearson_corr, _ = pearsonr(true_movie_vector, predicted_movie_vector)

    threshold = np.mean(true_movie_vector)
    binarized_predicted = np.where(predicted_movie_vector > threshold, 1, 0)
    binarized_true = np.where(true_movie_vector > threshold, 1, 0)

    precision = precision_score(binarized_true, binarized_predicted,
                                zero_division=0)
    recall = recall_score(binarized_true, binarized_predicted, zero_division=0)
    f1 = f1_score(binarized_true, binarized_predicted, zero_division=0)

    return mse, mae, pearson_corr, precision, recall, f1

# Function for TF-IDF recommendations
def recommend(movie):

```

```

print('          ')
print('TF-IDF:')
movie_index = new_df[new_df['title'] == movie].index[0]
distances = similarity[movie_index]
movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda
x: x[1])[1:6]

for i in movies_list:
    print(new_df.iloc[i[0]].title)

    mse,    mae,    pearson_corr,    precision,    recall,    f1    =
evaluate_recommendations(movie, [i[0] for i in movies_list], vectors)
    print("          ")
    print("Mean Squared Error:", mse)
    print("Mean Absolute Error:", mae)

# Function for CountVectorizer recommendations
def recommend2(movie):
    print('          ')
    print('CountVectorizer:')
    movie_index = new_df[new_df['title'] == movie].index[0]
    distances = similarity2[movie_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda
x: x[1])[1:6]

    for i in movies_list:
        print(new_df.iloc[i[0]].title)

        mse,    mae,    pearson_corr,    precision,    recall,    f1    =
evaluate_recommendations(movie, [i[0] for i in movies_list], vectors2)
        print("          ")
        print("Mean Squared Error:", mse)
        print("Mean Absolute Error:", mae)

# Function for KNN recommendations
def recommend_knn(movie):
    print('          ')
    print('KNN:')
    movie_id = new_df[new_df['title'] == movie].index[0]
    movie_vec = knn_vectors[movie_id]
    distances, indices = model_knn.kneighbors(movie_vec.reshape(1, -1),
n_neighbors=6)

```

```

recommended_indices = indices[0][1:6] # Take the next 5 recommendations

recommendations = new_df['title'].iloc[recommended_indices]

for title in recommendations:
    print(title)

    mse,    mae,    pearson_corr,    precision,    recall,    f1    =
evaluate_recommendations(movie, recommended_indices, knn_vectors)
    print("      ")
    print("Mean Squared Error:", mse)
    print("Mean Absolute Error:", mae)

recommendationBasedMovie = 'Cars'
recommend(recommendationBasedMovie)
recommend2(recommendationBasedMovie)
recommend_knn(recommendationBasedMovie)

```

ДОДАТОК Б

«Розробка рекомендаційної системи з використанням методів машинного навчання та статистичного аналізу»

Виконав:

Бородатюк Дмитро Михайлович

Науковий керівник:

асистент Соболь Ольга Олександрівна

Вступ

- Об'єкт дослідження: процес генерації рекомендацій на прикладі кінофільмів.
- Предмет дослідження: методи машинного навчання та статистичного аналізу і використання їх в задачах генерації рекомендацій.
- Мета дослідження: дослідження та порівняння різних методів та підходів до створення рекомендаційних систем та розробка програмного продукту на прикладі відеострімінгового сервісу.

Актуальність роботи

Сфери використання рекомендаційних систем:

- Застосунки (AppStore)
- Платформи електронної комерції (Amazon, Rozetka)
- Відео/фільми (YouTube, Netflix)
- Музика (Spotify, YouTube Music)

Переваги для бізнесу:

- Підвищення дохідності: було створено багато алгоритмів, як наприклад на сайті Amazon, де використовують рекомендаційні системи для товарів, щоб збільшити дохід на 35% за рахунок збільшення кількості продажів.
- Задоволеність клієнтів: клієнти очікують побачити рекомендації схожих продуктів на основі їхнього останнього пошуку на сайті. Задоволеність клієнтів сприяє їхньому утриманню.
- Персоналізація: зібрані дані використовуються, щоб покращити загальні послуги вебсайту та переконатися, що вони підходять відповідно до уподобань користувача.



Постановка задачі

Аналіз існуючих методів генерації рекомендацій

Аналіз метрик оцінювання якості надання рекомендацій

Розробка архітектури рекомендаційної системи на прикладі відеострімінгового сервісу

Аналіз вхідних даних

Підготовка даних

Розробка методів рекомендації

Аналіз результатів

Розробка інтерфейсу для інтеграції найкращого методу

Основні типи рекомендаційних систем

Системи на основі вмісту

Не використовують історію оцінок інших користувачів. Замість цього вони спираються на опис елементів контенту, представлений у вигляді ключових слів, тегів, категорій або інших метаданих. Аналізуючи ці дані, система виявляє зв'язки та подібності між елементами, формуючи на їх основі рекомендації для конкретного користувача.

Системи на основі колаборативного фільтрування

Будують свої рекомендації на основі вподобань інших користувачів. Ключовою концепцією цих систем є те, що для формування рекомендації приймається до уваги поведінка інших користувачів.

Гібридні системи

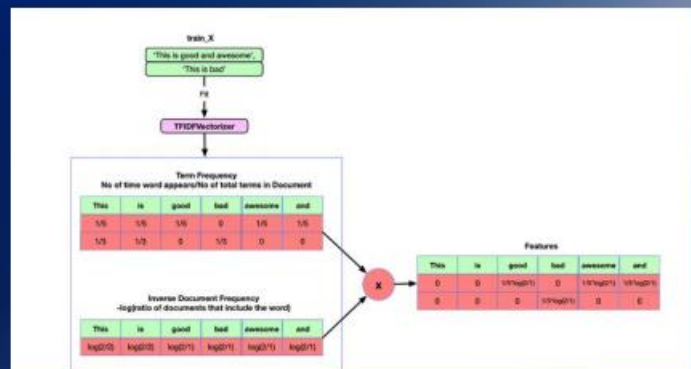
Поеднують різні існуючі моделі, такі як фільтрація на основі вмісту, колаборативна фільтрація або будь-яку іншу техніку рекомендації. Можуть бути реалізовані шляхом окремого формування прогнозів на основі вмісту та на основі колаборативного підходу, а потім їх комбінуванням, або шляхом об'єднання підходів у єдину модель.



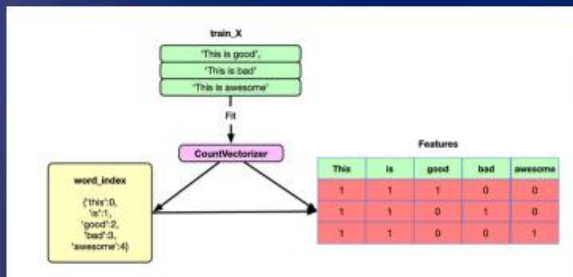
Методи, використані для побудови рекомендаційної системи

TF-IDF + Косинусна Схожість

Головна ідея полягає в тому, щоб оцінити важливість кожного слова в документі та виміряти схожість між документами. TF-IDF (Term Frequency-Inverse Document Frequency) перетворює текстові дані у вектори, де кожне слово має вагу, що залежить від його частоти в документі та рідкості у всьому корпусі. Після цього косинусна схожість вимірює кут між векторами, дозволяючи знайти схожі документи. Цей метод ефективний для визначення подібності текстів і знаходження найближчих сусідів.



Методи, використані для побудови рекомендаційної системи



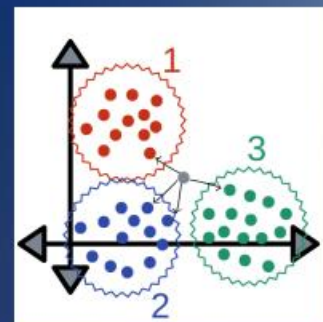
CountVectorizer + Косинусна Схожість

Головна ідея полягає в підрахунку частоти входжень кожного слова в документі і використанні цих частот для вимірювання схожості між документами. CountVectorizer перетворює текст у вектори частот, а косинусна схожість вимірює схожість між цими векторами. Це простий та швидкий метод для обробки текстових даних, який дозволяє знаходити схожі документи на основі їхнього змісту.

Методи, використані для побудови рекомендаційної системи

KNN (K-Nearest Neighbors)

Головна ідея полягає в знаходженні найближчих сусідів на основі обчислених відстаней між векторами даних. KNN (K-Nearest Neighbors) використовує відстань між векторними представленнями, щоб знайти k найближчих сусідів для кожного об'єкта. Це простий, швидкий і портативний алгоритм, який підтримує як регресію, так і класифікацію, дозволяючи знаходити схожі об'єкти (наприклад, фільми) на основі їхніх характеристик.



Основні критерії якості генерації рекомендації

Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^n (r_i - \hat{r}_i)^2$$

Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum_{i=1}^n |r_i - \hat{r}_i|$$

Precision, Recall та F1-Score

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Основні критерії якості генерації рекомендації

- MSE вимірює середньоквадратичну помилку між прогнозованими і фактичними значеннями, тому менші значення MSE вказують на кращу точність моделі рекомендаційної системи.
- MAE є корисним показником для оцінки середньої величини помилок в прогнозах, незалежно від напрямку помилки. MAE вимірює середнє абсолютне значення відхилень між спостережуваними та передбаченими значеннями.
- Precision вимірює частку правильних рекомендацій серед усіх наданих рекомендацій
- Recall вимірює частку правильно рекомендованих елементів серед усіх релевантних елементів
- F1-Score є гармонійним середнім значенням Precision та Recall, що забезпечує баланс між цими двома метриками

Вхідні дані

Данні були взяті з публічного веб-сайту Kaggle, де їх було надано сервісом The Movie Data Base (TMDB). Сервіс TMDB надав вільний доступ до даних про 5000 найпопулярніших фільмів у форматі двох датасетів формату csv.

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

Блок-схема реалізації продукту



Робота з даними

Було усунуто проблеми з пропущеними даними, а також були проведені наступні маніпуляції:

- Створені поля метаданих з полів ознак, на основі яких робляться рекомендації;
- Переведені дані формату об'єкт до формату тексту;
- Переведені всі слова до нижнього регістру;
- За допомогою алгоритму Porter Stemmer слова зведені до єдиної форми без закінчень;

Результати різних методів

Результати роботи програмного продукту для фільму Titanic:

TF-IDF:

```
recommend('Titanic')
✓ 0.0s
```

Raise the Titanic
Ghost Ship
Poseidon
In the Heart of the Sea
Pirates of the Caribbean: On Stranger Tides

Mean Squared Error: 0.0002843955457007855
Mean Absolute Error: 0.0015248377944807585

CountVectorizer:

```
recommend2('Titanic')
✓ 0.0s
```

The Notebook
Under the Same Moon
Ghost Ship
The Bounty
Pirates of the Caribbean: On Stranger Tides

Mean Squared Error: 0.0204
Mean Absolute Error: 0.0156

KNN:

```
recommend_knn('Titanic')
✓ 0.3s
```

KNN:
The Boys from Brazil
Year One
Spaceballs
Sphinx
Patton

Mean Squared Error: 0.00039616872129564297
Mean Absolute Error: 0.0017597664291045991

Результати метрик

	TF-IDF		CountVectorizer		KNN	
	MSE	MAE	MSE	MAE	MSE	MAE
Titanic	0.0002844	0.001525	0.0204	0.0156	0.000396	0.001759
Cars	0.00025392	0.00156	0.0108	0.01	0.000398	0.002239
Moneyball	0.0002745	0.0013781	0.0084	0.008	0.000398	0.001598
Inception	0.0003325	0.00188	0.0142	0.0126	0.0004	0.001798

Приклад роботи графічного інтерфейсу

Movie recommender

Select a movie

Unlabeled input

No results

Movie recommender

Select a movie

Titanic

Recommend the Titanic

Remember the Titanic

Crash of the Titans

Awake of the Titanic

The Chambermaid on the Titanic

Movie recommender

Select a movie

Titanic

Recommend

Remember the Titanic

Crash of the Titans

Awake of the Titanic

The Chambermaid on the Titanic

Overview: To obtain a supply of a rare mineral, a ship carrying slaves is hijacked by a pirate ship on the Bering Sea. The pirates claim the vessel as their own. Once they begin using the giant ship towards their, a series of events occur which happen and the group becomes trapped inside the ship, which they eventually are released by a dramatic crash.

Release Date: 2003-08-01

Overview: A passenger ship carrying slaves is hijacked by a pirate ship on the Bering Sea. The pirates claim the vessel as their own. Once they begin using the giant ship towards their, a series of events occur which happen and the group becomes trapped inside the ship, which they eventually are released by a dramatic crash.

Release Date: 2003-08-01

Overview: A passenger ship carrying slaves is hijacked by a pirate ship on the Bering Sea. The pirates claim the vessel as their own. Once they begin using the giant ship towards their, a series of events occur which happen and the group becomes trapped inside the ship, which they eventually are released by a dramatic crash.

Release Date: 2003-08-01

Overview: A passenger ship carrying slaves is hijacked by a pirate ship on the Bering Sea. The pirates claim the vessel as their own. Once they begin using the giant ship towards their, a series of events occur which happen and the group becomes trapped inside the ship, which they eventually are released by a dramatic crash.

Release Date: 2003-08-01

Overview: A passenger ship carrying slaves is hijacked by a pirate ship on the Bering Sea. The pirates claim the vessel as their own. Once they begin using the giant ship towards their, a series of events occur which happen and the group becomes trapped inside the ship, which they eventually are released by a dramatic crash.

Release Date: 2003-08-01

Висновки

- Виконано порівняльний аналіз відомих рішень генерації рекомендацій, їх типи та методи
- Визначено метрики оцінювання якості рекомендації
- Виконано аналіз та обробку вхідних даних
- Реалізовано програмно обрані моделі надання рекомендації
- Обрано найкращі моделі за метриками
- Побудовано графічний інтерфейс програми

Подальші дослідження

- Створити динамічний датасет з періодичним оновленням з метою розширення списку фільмів, які можна рекомендувати задля підвищення точності рекомендації
- Розглянути інші методи для вирішення даної задачі, наприклад рекомендація на основі нейронної мережі
- Подальший розвиток інтерфейсу

Дякую за увагу!