Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

Студента Говорухи Андрія Юрійовича

академічної групи 124м-23-1

спеціальності 124 Системний аналіз

на тему: «Розробка рекомендаційної системи вибору фільмів для онлайн кінотеатру.»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*к.т.н., доц. Мінєєв О.С.*			
розділів:				
Інформаційно-аналітичний	*к.т.н., доц. Мінєєв О.С.*			
Спеціальний розділ	*к.т.н., доц. Мінєєв О.С..*			
Рецензент				
Нормоконтролер	*к.ф.-м.н., доц. Хом'як Т.В.*			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу та управління
(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище,ініціали)

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *магістра*

студенту *Говорусі А.Ю.* **академічної групи** *124м- 23-1*
спеціальності: *124 Системний аналіз*
на тему «*Розробка рекомендаційної системи вибору фільмів для онлайн кінотеатру.»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від 26.10.2024 р. №275 – С

Розділ	Зміст	Терміни виконання
1. Інформаційно-аналітичний розділ	*Визначити предметну область дослідження та проблему, що розв'язується. Обґрунтувати методи для виконання поставлених завдань.*	10.09.2024 – 01.11.2024
2.Спеціальний розділ	*Розв'язати поставлені задачі: описати структуру об'єкта досліджень, розробити рекомендаційну систему вибору фільмів.*	01.11.2024 – 30.12.2024

Завдання видано ___ *доц. Мінєєв О.С.*
(підпис) (прізвище, ініціали)

Дата видачі: *06.09.2024 р.*
Дата подання до екзаменаційної комісії: *26.12.2024 р.*

Прийнято до виконання _____ Говоруха А.Ю.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 96 сторінки, 11 таблиць, 44 рисунків, 9 додатків, 21 джерел.

Зі зростанням кількості доступного контенту, користувачам стає дедалі важче знайти те, що їх дійсно цікавить. Традиційні методи пошуку не завжди задовольняють потреби користувачів, оскільки вони можуть бути занадто загальними і не персоналізованими. Відповідно, існує потреба у створенні рекомендаційної системи, яка здатна аналізувати великі обсяги даних та надавати релевантні та персоналізовані пропозиції.

Об'єкт дослідження: результати діяльності незалежного сайту «MovieLens», який збирає анонімні відгуки про фільми, за період з 2019 по 2023 рік.

Мета роботи: підвищення ефективності роботи онлайн-кінотеатру за рахунок створення рекомендаційної системи підбору фільмів.

Предмет дослідження: розробка автоматизованої рекомендаційної системи підбору фільмів для користувачів онлайн-кінотеатру.

В *інформаційно-аналітичному розділі* розглянуто відомості про стратегії роботи рекомендаційних систем, особливості обробки природньої мови (NLP), моделі розрахунку рейтингу, які навчаються за допомогою машинного навчання.

У *спеціальному розділі* описано структуру об'єкта і алгоритми запропонованих систем, виконано підготовку даних до розробки рекомендаційних систем, порівняно результати роботи моделей.

Практична цінність роботи полягає в покращені досвіду взаємодії користувача з онлайн-кінотеатром, виявлення інформації про користувача, що може допомогти виявити проблеми та загрози і вирішити їх завчасно.

РЕКОМЕНДАЦІЙНА СИСТЕМА, КОРИСТУВАЧ, ВІДГУК, ОЦІНКА, ТЕГИ, ФІЛЬМ, SVD, МАТРИЧНА ФАКТОРИЗАЦІЯ, ПРИРОДНЯ МОВА, МАШИНЕ НАВЧАННЯ, МОДЕЛЬ, GLOVE

ABSTRACT

Explanatory note: 96 pages, 11 tables, 44 drawings, 9 apps, 21 sources.

With the increasing amount of available content, it is becoming increasingly difficult for users to find what they are really interested in. Traditional search methods do not always meet users' needs, as they can be too general and not personalized. Accordingly, there is a need to create a recommender system that is able to analyze large amounts of data and provide relevant and personalized suggestions.

Object of study: results of the activities of the independent website "MovieLens", which collects anonymous film reviews, for the period from 2019 to 2023.

Objective: Increasing the efficiency of the online cinema by creating a recommender system for selecting films.

Subject of research: Creation of an automated recommender system for selecting films for users of online-cinema.

In the information-analytical section discussed information about the strategies of recommender systems, features of natural language processing (NLP), and rating calculation models that are trained using machine learning.

Special section describes the structure of the object and the algorithms of the proposed systems , preparing data for the development of recommender systems, and comparison of the results of the models.

The practical value of the work consists in improving the user's interaction experience with the online-cinema, exploring the information about the user, which can help identify problems and allow it to be used in solving the identified issues.

RECOMMENDER SYSTEM, USER, FEEDBACK, RATING, TAGS, MOVIE, SVD, MATRIX FACTORIZATION, NATURAL LANGUAGE, MACHINE LEARNING, MODEL, GLOVE

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ — Штучний інтелект

ML — Machine Learning

NLP — Natural language processing

kNN — k-Nearest Neighbour

НМ — Нейронна мережа

Word2vec — Word to Vector

GloVe — Global Vectors

OHE — One-hot-encoding

SVD — Singular Value Decomposition

IMDb — Internet Movie Database

RMSE — Root Mean Squared Error

MF — Matrix Factorization

ВСТУП

Рекомендаційні системи стали невід'ємною частиною сучасного цифрового світу, впливаючи на наші рішення у виборі контенту, товарів та послуг. Однією з найбільш популярних і корисних областей їх застосування є індустрія кіно та потокових сервісів. Завдяки рекомендаційним системам, глядачі можуть знайти фільми та серіали, які найбільше відповідають їхнім смакам та інтересам.

Метою цієї роботи є підвищення ефективності роботи онлайн-кінотеатру за рахунок створення рекомендаційної системи підбору фільмів.

Актуальність вирішення цієї задачі полягає у розробці засобів для автоматизованої обробки контенту, обсяг якого постійно зростає; необхідності персоналізації користувацького досвіду, підвищенні задоволеності користувачів, яка приведе до збільшення коефіцієнту утримання аудиторії і, як наслідок, економічних вигодах для онлайн-кінотеатру.

Об’єктом дослідження є результати діяльності незалежного сайту «MovieLens», який збирає анонімні відгуки про фільми, за період з 2019 по 2023 рік.

Предметом дослідження є розробка автоматизованої рекомендаційної системи підбору фільмів для користувачів онлайн-кінотеатру.

Методи дослідження. Методологічною основою магістерської роботи є сукупність методів і прийомів пізнання і дослідження. Вибір методів дослідження обумовлений особливостями об’єкта та предмета, а також метою та завданнями дослідження. У роботі використані такі методи:

Аналіз літературних джерел — вивчення наукової і технічної літератури, статей та інших джерел, пов'язаних з темою дослідження.

Емпіричні методи — спостереження, експерименти, опитування та інтерв'ю для збору первинних даних.

Метод моделювання — створення моделей, що дозволяють імітувати реальні процеси і явища для подальшого аналізу.

Статистичні методи — обробка і аналіз даних за допомогою статистичних методів, таких як регресійний аналіз, кореляційний аналіз тощо.

Метод комп'ютерного моделювання — використання комп'ютерних програм і алгоритмів для моделювання і аналізу складних систем.

Ці методи забезпечують комплексний підхід до дослідження і дозволяють отримати об'єктивні та науково обґрунтовані результати.

Теоретичне значення Для вирішення практичного завдання, поставленого цією кваліфікаційною роботою, було запропоновано поєднання моделі Funk SVD з ембедингами тегів користувачів і фільмів.

Наукова новизна отриманих результатів. Поєднання моделі Funk SVD з ембедингами тегів користувачів і фільмів дозволило значно покращити традиційну модель Funk SVD і зробити рекомендації більш персоналізованими, що суттєво підвищує ефективність рекомендаційних систем.

Практичне значення отриманих результатів. Розроблена система може бути використана для підвищення якості рекомендацій у різних сервісах потокових відео, забезпечуючи користувачам більш точні та релевантні рекомендації, що збільшує їх задоволеність і утримує на платформі.

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, двох розділів, висновків, списку літератури з 21 найменувань та 9 додатків. Загальний обсяг кваліфікаційної роботи складає 96 сторінки, з них 67 сторінки основного тексту, який містить 44 рисунків та 11 таблиць.

Виконання роботи буде поділено на слідуючі етапи:

- провести аналіз науково-технічних публікацій та даних Інтернет джерел в області сучасних підходів для обробки природномовних текстів;

- провести порівняльний аналіз основних відомих методів для вирішення задачі класифікації;

- здійснити вибір моделі та методів для проведення дослідження;

- здійснити вибір набору даних для експериментальних досліджень;

- підготовка даних для безперешкодної роботи моделі;

- розробити програмне забезпечення для запропонованих моделей;

- провести обґрунтований вибір моделі за допомогою метрик якості;

- здійснити дослідження можливостей реалізованого програмного додатку.

1. ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Загальні відомості

У якості предметної області було обрано задачу побудови рекомендаційної системи підбору фільмів для онлайн кінотеатру. Рекомендаційна система – це комплекс алгоритмів, програм і сервісів, чия задача спрогнозувати список предметів, які можуть зацікавити користувача під час знаходження на сайті.

Яскравим прикладом такої системи можна побачити на відеохостингу Youtube. Якщо користувач деякий час користувався сайтом, при цьому авторизувавшись на ньому, то він, сайт, знає, які нові відео можна запропонувати такому користувачу, тому що сайт знає, що ви дивились в минулому. Інший приклад, якщо користувач неавторизований, то сайт пропонує користувачу зробити пошук відео, яке його цікавить. Такий випадок називається проблема холодного старту (cold-start problem).

Засобом, за допомогою якого визначають список рекомендованих предметів, може бути матриця взаємодії користувача з предметами або матриці, які містить в собі дані користувача і дані про предмет . Ці дані можуть бути представлені у вигляді чисел, тексту, дат.

У таблиці 1.1 зображено приклад матриці взаємодії.

Таблиця 1.1

Матриця взаємодії користувача і предмету.

	item 1	item 2	item 3	item 4	item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

У комірках знаходиться результат взаємодії або рейтинг користувача стосовно деякого предмету. Комірки з нулями означають відсутність взаємодії, а значення в інших комірках можуть відрізнятись, наприклад, вони можуть бути записані у вигляді: «-1» - не сподобався предмет, «1» - сподобався ; полуінтервал (0;1]; рейтингові шкали (5 бальні, 10 бальні тощо). [1]

На рисунку 1.1 зображено приклад матриць ознак користувача і предметів. У комірках знаходиться міра відношення деякої ознаки *feature_i* до користувача або предмету. Міра може бути бінарною або дійсним числом.

	feature 1	feature 2	feature 3	feature 4
user 1	0	1	1	0
user 2	1	0	0	1
user 3	1	1	1	0
user 4	1	0	1	0
user 5	1	1	1	0

	feature 1	feature 2	feature 3	feature 4
item 1	0	1	0	0
item 2	1	0	1	0
item 3	0	1	1	0
item 4	0	1	1	0
item 5	0	0	1	1

	feature 1	feature 2	feature 3	feature 4
user 1	0,989773	0,879935	0,894329	0,791026
user 2	0,57779	0,925307	0,893068	0,681992
user 3	0,815022	0,622627	0,606321	0,807646
user 4	0,428872	0,000579	0,59366	0,377366
user 5	0,520798	0,388522	0,96698	0,730144

	feature 1	feature 2	feature 3	feature 4
item 1	0,991546	0,085481	0,291769	0,725655
item 2	0,039399	0,137216	0,327148	0,624676
item 3	0,358138	0,521522	0,309891	0,229317
item 4	0,000365	0,246084	0,949662	0,048483
item 5	0,946535	0,725962	0,723832	0,659038

Рисунок 1.1 Матриця ознак користувача і предметів.

Отримана інформація відрізняється між собою за будовою – десь лише одна матриця взаємодії, а десь наявні матриці ознак. Тому для різних матриць використовуються свої стратегії, якою керується модель, яка вирішує задачу рекомендації. Поширеними стратегіями є: колаборативна фільтрація (collaborative filtering), фільтрація на основі вмісту (content-based filtering), гібридна, яка поєднують ці два методи.

1.2. Стратегії рекомендаційної системи

У цьому підрозділі буде згадуватись таке поняття як міра схожості. Це поняття містить в собі величину схожості користувача з товаром за умови, що їх ознаки мають однаковий сенс. Тобто, якщо перша ознака користувача відповідає жанру «Драма», то і для товару ця ознака повинна відповідати цьому жанру – і так для кожної ознаки. Інакше сенс такого порівняння зникає.

1.2.1 Міра схожості

Існує три поширені формули для розрахунку схожості векторів дійсних значень: евклідова відстань, косинусна схожість, .

1. Евклідова відстань

Найбільш поширеною та інтуїтивно зрозумілою метрикою відстані є евклідова відстань. Ми можемо уявити це як кількість простору між двома об’єктами даних. Наприклад, наскільки далеко ваш екран знаходиться від вашого обличчя. [2]

Запишемо формулу евклідової відстані:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}, \quad (1.1)$$

де q, p – вектор значень.

Евклідова відстань чутлива як до напрямку, так і до величини векторів, що робить її придатною для сценаріїв, де абсолютні позиції у векторному просторі мають значення, ця чутливість може бути як перевагою, так і недоліком залежно від застосування. [2]

2. Косинусна схожість

Термін «косинусна схожість» використовується для позначення різниці між орієнтацією двох векторів, а саме вона вимірює кут між двома векторами.

Запишемо формулу косинусної схожості:

$$cos(\theta) = \frac{A*B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \quad (1.2)$$

де A, B – вектор значень.

Важливою властивістю косинусної подібності є те, що вона не знаходиться під впливом масштабування, тобто враховує лише кут між векторами, а не їх величини, що особливо корисно під час порівняння документів або вставлених слів різної довжини. Якщо ви працюєте з нормалізованими векторами, косинусна схожість еквівалентна скалярному добутку. [2]

3. Скалярний добуток

Скалярний добуток — це проекція одного вектора на інший. Чим більший кут між двома векторами, тим менший скалярний добуток. Він також масштабується з довжиною меншого вектора. Отже, ми використовуємо скалярний добуток, коли для нас має значення орієнтація та відстань. [2]

Запишемо формулу скалярного добутку:

$$a \cdot b = \sum_{i=1}^{n} a_i * b_i \ , \quad (1.3)$$

де a, b – вектор значень.

Скалярний добуток є щось середнє між евклідовою відстанню та косинусною схожістю. Коли мова заходить про нормалізовані значення векторів, то результат збігається з косинусною схожістю. Тому скалярний добуток може підійти як для великих наборів даних, так і для нормалізованих чи не нормалізованих наборів даних. [2]

У цьому підрозділі ми дізналися про три найкорисніші метрики визначення векторної подібності: евклідова відстань, косинусна схожість і схожість за скалярним добутком. Кожна із них має різні випадки

використання. Евклідова – це коли нам важлива різниця величин. Косинусна — це коли нам важлива різниця в орієнтації. Скалярний добуток – це коли ми дбаємо про різницю в величині та орієнтації.[2]

1.2.2 Фільтрація на основі вмісту

Система рекомендацій на основі вмісту є популярним і широко використовуваним підходом для надання персоналізованих рекомендацій користувачам. Ця система працює таким чином, що вподобання користувача можна передбачити на основі його попередньої взаємодії з товарами, наприклад, історія переглядів і покупок. Таким чином, система буде рекомендувати користувачеві елементи, схожі на елементи, з якими він раніше взаємодіяв.

Для того, щоб ця система працювала, товар повинен мати ознаки, які можуть бути перенесені на користувача. Розглянемо алгоритм роботи цієї стратегії:

1. Обрати користувача, для якого треба зробити прогноз. У таблиці 1.2 можна побачити рейтинги, які залишав цей користувач для товарів.

Таблиця 1.2

Рейтинги користувача

	item 1	item 2	item 3	item 4	item 5
user 1	0	5	0	2	4

У таблиці 1.3 можна побачити ознаки для кожного товару.

Таблиця 1.3

Ознаки товарів

	feature 1	feature 2	feature 3	feature 4
item 1	0	1	1	1
item 2	1	1	1	0
item 3	0	1	0	1
item 4	1	0	0	0
item 5	1	1	1	0

2. Виписати матрицю ознак, де на місці наявної ознаки знаходиться значення рейтингу відповідного товару. (див. таблицю 1.4)

Таблиця 1.4

Матриця ознак.

	feature 1	feature 2	feature 3	feature 4
item 1	0	0	0	0
item 2	5	5	5	0
item 3	0	0	0	0
item 4	2	0	0	0
item 5	4	4	4	0

3. Вивести ознаки користувача, поділивши суму по ознакам на кількість товарів помножене на максимально можливий рейтинг. Після того, як вивели ознаки, нормалізуємо їх, поділивши кожне значення на суму ознак.

	feature_1	feature_2	feature_3	feature_4
Item 1	0	0	0	0
Item 2	5	5	5	0
Item 3	0	0	0	0
Item 4	2	0	0	0
Item 5	4	4	4	0
user_features	0,44	0,36	0,36	0
user_features normalized	0,37931	0,31034	0,31034	0

Рисунок 1.2. Ознаки користувача.

4. Розрахувавши ознаки користувача, тепер ми можемо розрахувати схожість з товарами, з якими він ще не взаємодіяв. Наприклад, як міру схожості можна взяти формулу скалярного множення. Тоді для користувача будуть розраховані наступні очікувані рейтинги.

	Скалярне множення	5 бальний рейтинг
Item 1	0,6207	3,1034
Item 2	1,0000	5,0000
Item 3	0,3103	1,5517
Item 4	0,3793	1,8966
Item 5	1,0000	5,0000

Рисунок 1.3. Очікувані рейтинги для користувача.

Виділені кольором рядки відносяться до товарів, з якими користувач вже взаємодіяв. Тоді залишаються товари 1 і 3, серед яких товар 1 має більший рейтинг. Це значить, що ми можемо рекомендувати його для користувача.

Як можна здогадатись, у такої системи ознаки користувача будуть схилені в бік до товарів, які були переглянуті користувачем. Якщо йому будуть рекомендувати товари за схемою розглянутою вище, це може стати проблемою для того, щоб побачити в рекомендаціях товар, який не схожий на попередні.

1.2.3 Колаборативна фільтрація

Один з підходів побудови рекомендаційної системи, який широко використовується – це колаборативна фільтрація. Вона базується на припущені того, що люди які погодились в минулому погодяться у майбутньому, а значить їм сподобаються предмети схожі на предмети, які їм подобались в минулому. Система генерує рекомендацію, використовуючи лише інформацію про вподобання для різних користувачів або предметів, іншими словами маючи лише матрицю взаємодії. Відповідно звідси випливають різновиди колаборативної фільтрації:

1. **На основі пам’яті (Memory-based)**: Системи побудовані на цьому алгоритмі використовують принцип ґрунтування на користувачі (user-

based) або предметі(item-based). На рисунку 1.4 проілюстровано принцип роботи алгоритму.

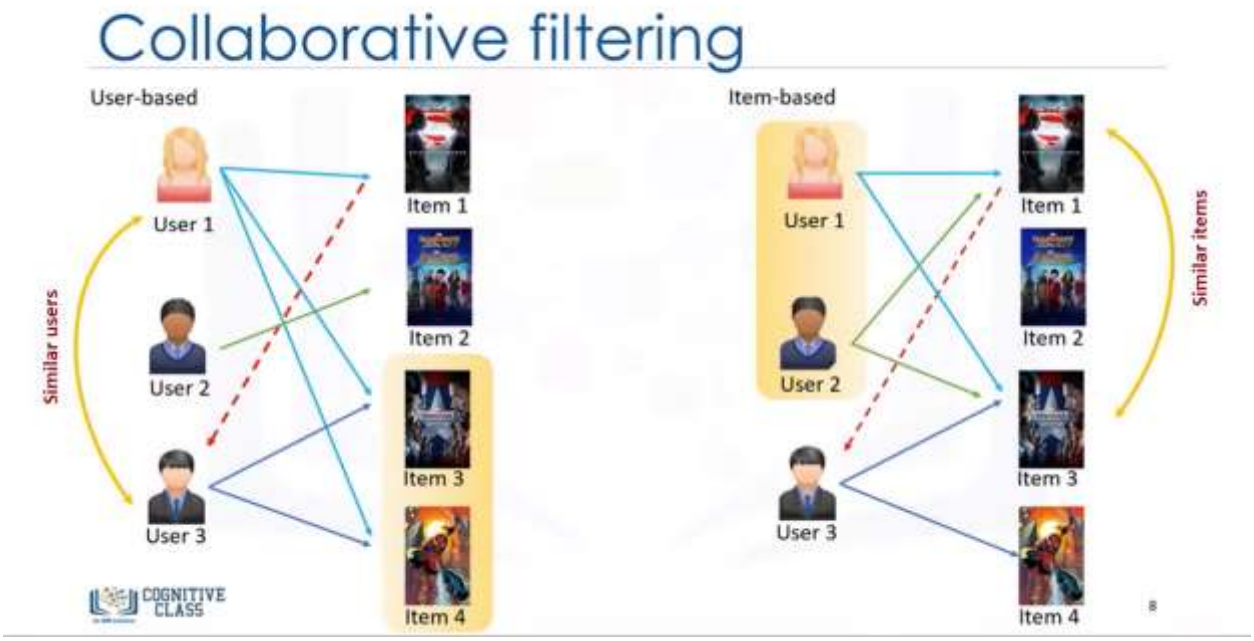


Рисунок 1.4. Алгоритм колаборативної фільтрації на основі користувачів (ліворуч) і предметів (праворуч).[3]

Розберемо кожен з алгоритмів окремо. Фільтрація на основі користувача полягає у тому, що для користувача знаходиться схожий користувач, а рекомендації обираються за принципом рекомендувати те, що ще не бачив.[1]

Алгоритм виглядає наступним чином:

1. Обрати потрібного користувача, для якого треба зробити рекомендацію.

2. Розрахувати величину схожості оцінок обраного користувача з іншими. Це буде вектор розмірністю n, де n – кількість користувачів. У таблиці 1.5 записано матрицю схожості користувачів, дані для розрахунку були взяти з матриці у таблиці 1.1.

Таблиця 1.5

Розрахована матриця схожості за косинусною мірою.

	user_1	user_2	user_3	user_4	user_5
user_1	1	0,23	0	0,353	0,84
user_2	0,23	1	0	0,46	0,711
user_3	0	0	1	0,686	0
user_4	0,353	0,46	0,686	1	0,545
user_5	0,84	0,711	0	0,545	1

3. Наприклад, для користувача 1 потрібно зробити рекомендацію. Можна побачити, що у векторі схожості для цього користувача найбільшу схожість має користувач 5.

4. Повернемось до матриці взаємодії і витягнемо рядки оцінок для користувача 1 і 5. Порівняємо їх і знайдемо товари, які будуть рекомендовані користувачу 1 (див. таблицю 1.6).

Таблиця 1.6

Вектори оцінок користувача 1 і 5

	item 1	item 2	item 3	item 4	item 5
user 1	0	3	0	3	4
user 2	4	3	0	4	4

5. Як можна побачити, користувач 1 не має оцінки для товару 1, тобто не взаємодіяв з ним, тому для нього буде рекомендовано цей предмет.

Перейдемо тепер до фільтрації базованої на предметі. Вона полягає у тому, що користувачу, якому сподобався товар 1 і був знайдений схожий на нього товар 2, буде рекомендовано товар 2.

Використання цієї методики можна побачити під час перегляду сторінок конкретного товару. Користувачу будуть рекомендувати товари схожі на той, який він зараз дивиться.

Алгоритм виглядає наступним чином:

1. Обрати предмет, для якого треба зробити рекомендацію.

2. Розрахувати величину схожості оцінок обраного товару з іншими. Це буде вектор розмірністю m, де m – кількість товарів. На рисунку

1.5 зображено матрицю схожості предметів, дані для розрахунку були взяти з матриці у таблиці 1.1.

	item_1	item_2	item_3	item_4	item_5
item_1	1	0,442	0,375	0,696	0,61
item_2	0,442	1	0	0,919	0,883
item_3	0,375	0	1	0	0,375
item_4	0,696	0,919	0	1	0,812
item_5	0,61	0,883	0,375	0,812	1

Рисунок 1.5. Розрахована матриця схожості за косинусною мірою.

3. Наприклад, для товар 2 потрібно зробити рекомендацію. Можна побачити, що у векторі схожості для цього предмету найбільшу схожість має товар 4.

4. Під час перегляду сторінки предмету 2, користувач може отримати в рекомендаціях товар 4.

Таким чином виконується рекомендації колаборативна фільтрація на основі пам’яті. Тепер розглянемо системи на основі моделі.

2. **На основі моделі (Model-based):** У цьому підході до реалізації рекомендаційної системи, модель колаборативної фільтрації розроблена за допомогою машинного навчання з метою прогнозувати результат взаємодії користувача з товаром. Такі моделі можна поділити на три підтипи: непараметричні, матрична факторизація, нейронні мережі.

До непараметричних моделей можна віднести моделі на основі кластеризації, наприклад, k Nearest Neighbors (kNN). Ідея кластеризації така ж, як і в системах на основі пам’яті. В алгоритмах на основі пам’яті ми використовуємо схожість оцінок між користувачами або товарами як вагові коефіцієнти для прогнозування оцінки для користувача та елемента. Різниця полягає в тому, що схожість в цьому підході обчислюються на основі моделі навчання без нагляду (unsupervised learning), а не за формулами схожості. У

цьому підході ми також обмежуємо кількість подібних користувачів як k, що робить систему більш масштабованою. [4]

Моделі матричної факторизації намагаються виконати декомпозицію матриці взаємодії. Іншими словами намагається розкласти одну матрицю на матриці меншого розміру. Наприклад, при розкладені на дві матриці перша матриця буде відповідати за ознаки користувача, а друга за ознаки товару. Ці ознаки можуть бути латентними (неявними), так і явними. [4]

Моделі побудовані за допомогою нейронних мереж можуть вирішити проблеми, які властиві матричній факторизації – наприклад, включення додаткових ознак користувача (країна, стать) або товару. За допомогою складної структури такі моделі можуть вивчати нелінійні зв’язки, на відмінну від матричного множення у матричній факторизації, яке не завжди може уловлювати складні зв’язки. Також мати проблему з тим, що використовуючи звичайне матричне множення може призвести до того, що всім користувачам будуть рекомендувати популярні предмети.[5]

1.2.4 Гібридні методи

Комбінація стратегій фільтрації на основі вмісту і колаборативної фільтрації називають гібридним методом рекомендації (Hybrid Methods). Такий підхід дозволить пом’якшити недоліки, які виникають при використані моделей окремо. Як було вказано, фільтрація на основі вмісту може бути схильна рекомендувати товари схожі один на одного, а колаборативна фільтрація може завищувати рекомендації для популярних товарів, не враховуючи ознаки користувача.

Колаборативна фільтрація і підхід на основі вмісту вважаються доповнюючими, оскільки колаборативна фільтрація рекомендує лише вже оцінені елементи, тим часом рекомендація на основі вмісту може рекомендувати нові елементи, ще не оцінені користувачем.

Щоб досягти кращих результатів, нам потрібен підхід, який має справу з різними слабкими сторонами індивідуальної техніки, використовує їхні

переваги та покладається на численні джерела, щоб використовувати найбільш відповідні. У цьому ключі була запропонована гібридизація двох типів фільтрації. Існує багато доступних методів гібридизації, таких як змішування, зважений підхід, комбінування ознак. Використовуючи ці методи, система поєднує різні підходи до рекомендацій в єдину гібридну систему рекомендацій.[6]

1.3 Обробка вхідних даних

У сьогоднішній час веб сайти намагаються отримувати як найбільше інформації про своїх користувачів для оптимізації роботи сайту. Ця інформація може складатись як з опису користувача, так і з метаданих – наприклад, пристрій, з якого він авторизувався, браузер тощо.

Інформація про фільми на сайті онлайн кінотеатру може містити дані про тривалість, жанри, акторський склад, бюджет, дату прем'єри тощо. Усі перераховані дані відрізняються між собою за типом: тривалість – числовий тип, жанр – текстовий, дата прем'єри – дата. Для того щоб мати змогу взаємодіяти з цими даними і знаходити корисні зв'язки між полями, використовуючи обчислювальні потужності комп'ютера, нам потрібно перевести всі дані в числовий формат.

На рисунку 1.6 зображено типовий набір даних про фільм.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
6	Heat (1995)	Action|Crime|Thriller
128954	I Love Sarah Jane (2008)	Horror|Romance
129021	Perez. (2014)	Crime|Drama|Thriller
129054	Love Camp (1977)	Adventure|Drama|Thriller

Рисунок 1.6. Типовий набір даних про фільм

Як можна побачити, дані максимально скомпоновані, наприклад, рік випуску фільму записано разом з назвою, жанри знаходяться в одній комірці і перелічені через розділовий знак.

Для того щоб не було непорозуміння даних треба відділити інформацію по різним коміркам, наприклад, рік випуску винести в окреме поле. Схожу операцію треба виконати з жанрами, де кожен з них представляє категорію. Для цього можна застосувати операцію One-Hot-Encoding (OHE).

1.3.1 One-hot-encoding

One-Hot-Encoding у машинному навчанні це перетворення категоріальної інформації у формат, який можна вводити в алгоритми машинного навчання для підвищення точності передбачення. Одночасне кодування є поширеним методом роботи з категоріальними даними в машинному навчанні.[7]

За допомогою цієї операції ми для кожного жанру зробимо окремий стовпчик, який буде зберігати «1», якщо цей жанр відноситься до фільму, і навпаки 0, якщо не відноситься. Таким чином ми отримаємо перетворений набір даних. На рисунку 1.7 можна побачити цей набір.

movieId	title	year	genres_adventure	genres_animation	genres_children	genres_comedy	genres_fantasy	genres_romance	genres_drama	genres_crime	genres_action	genres_thriller	genres_horror
1	Toy Story	1995	1	1	1	1	1	0	0	0	0	0	0
2	Jumanji	1995	1	0	1	0	1	0	0	0	0	0	0
4	Waiting to Exhale	1995	0	0	0	1	0	1	1	0	0	0	0
6	Heat	1995	0	0	0	0	0	0	0	1	1	1	0
128954	I Love Sarah Jane	2008	0	0	0	0	0	1	0	0	0	0	1
129021	Perez.	2014	0	0	0	0	0	0	1	1	0	1	0
129054	Love Camp	1977	1	0	0	0	0	0	1	0	0	1	0

Рисунок 1.7. Трансформований набір даних про фільм.

Як можна побачити, тепер більша частина інформації про фільм знаходиться у числовому форматі. Тепер розглянемо стовпчик значень року. Як можна побачити, різниця між значеннями може сягати і 37 одиниць, а частка від середнього складає всього 0.018. Тобто невелика зміна у великому числі може бути малопомітною під час використання цього набору даних.

Для вирішення цієї проблеми можна розбити дані на десятиріччя: 1990-ті, 2000-ті, 2010-ті тощо. Як можна зрозуміти, ми перевели дані з числового формату в категоріальний. Тоді ми отримаємо наступний набір (див. рисунок 1.8):

movieId	title	year	genres_adventure	genres_animation	genres_children	genres_comedy	genres_fantasy	genres_romance	genres_drama	genres_crime	genres_action	genres_thriller	genres_horror
1	Toy Story	1990s	1	1	1	1	1	0	0	0	0	0	0
2	Jumanji	1990s	1	0	1	0	1	0	0	0	0	0	0
4	Waiting to Exhale	1990s	0	0	0	1	0	1	1	0	0	0	0
6	Heat	1990s	0	0	0	0	0	0	0	1	1	1	0
128954	I Love Sarah Jane	2000s	0	0	0	0	0	1	0	0	0	0	1
129021	Perez.	2010s	0	0	0	0	0	0	1	1	0	1	0
129054	Love Camp	1970s	1	0	0	0	0	0	1	0	0	1	0

Рисунок 1.8 Трансформований набір даних про фільм.

Якщо тепер рік представлено як категорії, тоді ми можемо застосувати вже знайомий One-Hot-Encoding. Для кожного десятиріччя ми створимо окремий стовпчик, який буде містити «1» або «0».

movieId	title	1970s	1990s	2000s	2010s
1	Toy Story	0	1	0	0
2	Jumanji	0	1	0	0
4	Waiting to Exhale	0	1	0	0
6	Heat	0	1	0	0
128954	I Love Sarah Jane	0	0	1	0
129021	Perez.	0	0	0	1
129054	Love Camp	1	0	0	0

Рисунок 1.9. Фрагмент трансформованого набору даних.

Після такого перетворення у наборі залишився один стовпчик, який має текстові значення – це стовпчик з назвою фільму. Назва може складатись з декількох слів, які ми можемо так само розкласти за допомогою one-hot-encoding. Цю техніку, розкладати набір значень у вектор нулей і одиниць, також використовують у такій галузі машинного навчання як Natural Language Processing (NLP).

Обробка природної мови (NLP) — це підгалузь комп’ютерних наук та штучного інтелекту (ШІ), яка використовує машинне навчання, щоб дозволити комп’ютерам розуміти людську мову та спілкуватися нею.[19]

Наприклад, ми маємо текст з n унікальних слів і для того, і після кодування кожного слова за допомогою one-hot-encoding, ми отримаємо n векторів розміром $1 \times n$, де $n - 1$ значень дорівнюють нулю. Наприклад, словник, який був створений за допомогою 2 мільярдів повідомлень з сайту Twitter, сягає 1.2 мільйонів унікальних слів. Такий великий словник, що має 1.2 мільйони унікальних слів, створює серйозні обчислювальні виклики під час обробки тексту, особливо при використанні one-hot-encoding. Кожне слово представляється як вектор довжиною 1.2 мільйони з одним значенням, яке дорівнює одиниці, і всіма іншими значеннями, що дорівнюють нулю. Це призводить до високої розмірності векторів, що ускладнює обчислення та вимагає значних обсягів пам'яті.

Замість цього часто використовуються більш ефективні методи кодування такі як: Word Embedding.

1.3.2 Word Embedding

Word Embedding — це метод представлення слів у вигляді векторів у багатовимірному просторі. Метою є зберегти семантичну схожість між словами, тобто слова зі схожим значенням будуть мати вектори, розташовані близько один до одного у векторному просторі. Семантична схожість обчислюється за допомогою косинусної схожості, яку було описано у розділі 1.2.1.

На рисунку 1.10 можна побачити приклад ембедінгу слова.

car	0	1	2	3	4	5	6	7	8	9
	0,018	-53,580	-51,620	13,481	-82,326	14,553	-4,443	-62,604	17,256	-54,983

Рисунок 1.10. Приклад ембедінгу слова «car» вектором розмірністю 10.

На рисунку 1.11 можна побачити набір слів схожих між собою за семантикою, але замість чисел координати закодовані кольором, де червоний колір відповідає за від'ємне число, а синє за позитивне. Інтенсивність кольору відповідає абсолютному розміру числа.

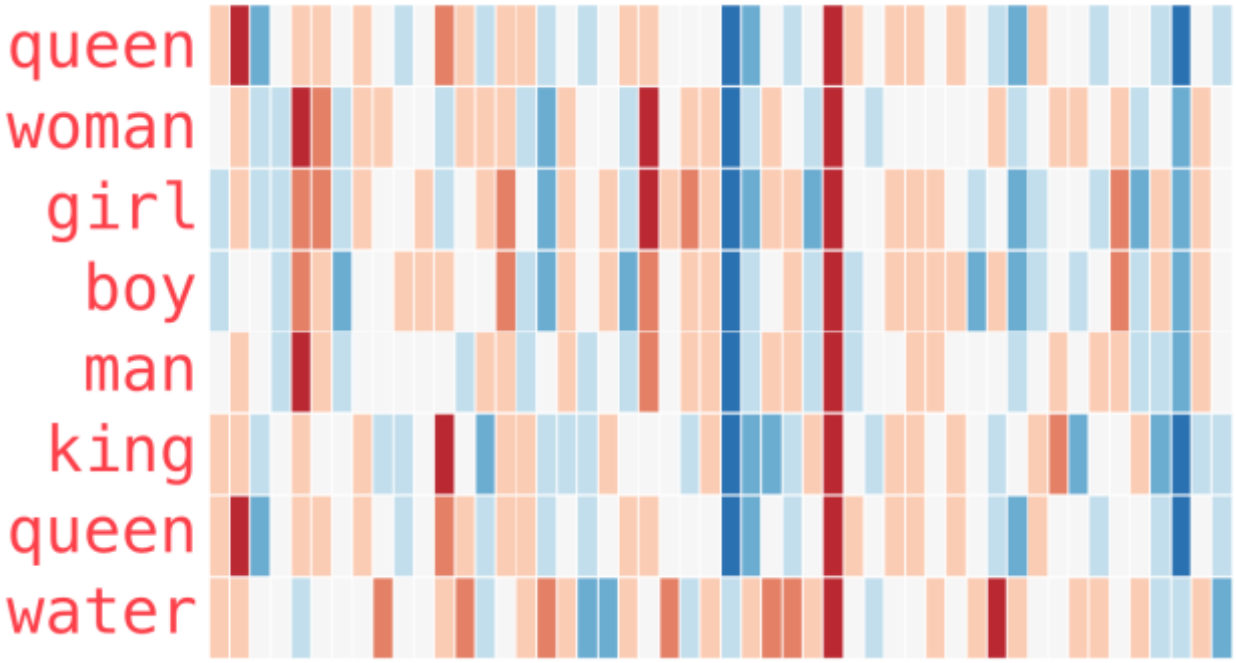


Рисунок 1.11. Приклад ембедінгу семантично схожих слів.

Як можна помітити, усі слова, які відносяться до людей мають синій стовпчик, а слово «water» відрізняється від інших. Можна припустити, що цей стовпчик відповідає за належність токену до людей.

Повернемося до самої ідеї заповнення дійсного простору значень. Результатом такого способу заповнення вектора з 10 параметрів є те, що окремому слову відповідає вектор, за допомогою якого можна розрахувати скалярну відстань, яка буде співпадати між словами одного роду або синонімами. У роботі word2vec [8] наведено наступний малюнок (див. рисунок 1.12), відстань між словами {“man”, “woman”}, {“uncle”, “aunt”}, {“king”,”queen”} приблизно однакова, що показує те, що модель, запропонована ним, може відрізняти слова антоніми, а різниця між підвидами цих антонімів буде приблизно однакова.

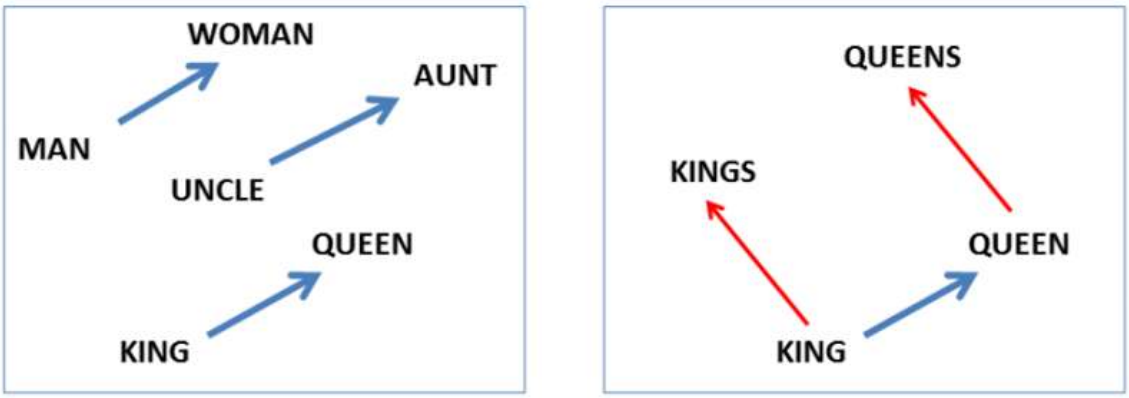


Рисунок 1.12. Приклад семантичного зв’язку токенів

Для того щоб створити такий набір слів виконуються наступні початкові кроки, які застосовуються для вирішення будь-яких завдань NLP – це сегментація та токенізація корпусу слів.

Корпус слів – це підібрана і опрацьована сукупність текстів, які використовуються як база для дослідження мови. Корпусом може бути набір відгуків або всі статті з сайту Wikipedia.[19]

Сегментація – розподіл тексту на абзаци, абзаців на речення. Іншими словами розбиття складної структури на декілька менш складних.[19]

Токенізація – перший крок при обробці речення. Він полягає в розбитті тексту на слова. Відповідно тепер кожне слово буде називатись *токеном*.[19]

Після цього ми можемо сформувати словник унікальних токенів корпусу і відредагувати координати векторів ембедингів. Для цієї розв’язання задачі існує 3 популярних рішення: word2vec, GloVe, fasttext. У цій кваліфікаційній роботі було використано метод GloVe.

1.3.3 GloVe

GloVe (Global Vectors for Word Representation) — це один з найбільш відомих методів, розроблений у Стенфордському університеті. GloVe поєднує в собі переваги локальних і глобальних методів представлення слів, що робить його потужним інструментом для різноманітних завдань NLP.

GloVe є алгоритмом для навчання векторних представлень слів, який використовує глобальні статистичні дані про їх спільну появу у великих текстових корпусах. Ідея полягає в тому, що частоти співпояви слів можуть розкрити семантичні зв'язки між ними. [9]

Процес починається зі створення матриці співпояви, де кожен елемент X_{ij} представляє кількість разів, коли слово i з'являється у контексті слова j. Ця матриця забезпечує глобальний огляд всіх відносин між словами в корпусі.

GloVe навчається так, щоб відношення між словами у векторному просторі відображало їх співпояву у текстах. Формула для навчання виглядає наступним чином:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \widetilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2, \qquad (1.4)$$

де V −розмір словника, X_{ij} − кількість співпояв слів i та j, $w_i, \widetilde{w}_j$ − вектори ембедінги слів i та j, $b_i, \tilde{b}_j$ − зміщення для ембедінгів слів i та j, $f\left(X_{ij}\right)$ − функція ваги, яка зменшує вплив рідких співпояв.

Ця формула дозволяє враховувати як локальні, так і глобальні контексти, що робить модель GloVe дуже ефективною.

Отже, повернемося до малюнку 1.3.4, замість використання one-hot-encoding, який займає велику кількість місця в матриці, ми можемо закодувати назву фільмів за допомогою Word Embedding. Для того, щоб не тренувати з нуля словник ембедингів, ми можемо завантажити один з готових словників англійської мови, в якому ми будемо робити пошук ембедингів токенів, наприклад, словник має розмір ембедингу 10 – кожен токен буде мати вектор розміром 10.

Знаходячи ембединг кожного токену назви, ми виконуємо операцію додавання векторів, що дасть ефект продемонстрований на рисунку 1.3.7, а саме знаходити семантичний зв’язок за допомогою арифметичних операцій

на векторами ембедингів. Отримані ембединги додамо в набір даних (див. рисунок 1.13).

movieId	title_embeddi ng_0	title_embeddi ng_1	title_embeddi ng_2	title_embeddi ng_3	title_embeddi ng_4	title_embeddi ng_5	title_embeddi ng_6	title_embeddi ng_7	title_embeddi ng_8	title_embeddi ng_9	1970s
1	-8,0889	32,1272	-58,1771	59,4202	81,7727	34,9287	51,8529	80,5206	19,9857	-17,1288	0
2	-76,3963	-77,2414	-54,1480	-26,0382	95,5918	79,5802	5,9488	29,9186	-92,7082	55,9287	0
4	14,2813	71,5845	13,5949	56,7552	-54,7637	67,4305	82,8629	68,3188	77,5409	-82,8883	0
6	-44,2150	16,6449	15,4320	-62,5162	-60,1098	-60,5495	78,2067	45,6028	-90,3349	81,5273	0
128954	67,0883	44,5792	62,6043	27,8848	-35,6839	-41,2675	35,9052	-60,0985	-61,0416	50,5519	0
129021	55,6245	-69,2198	-79,2624	-85,6956	84,7292	96,9987	48,3508	12,8022	-39,6807	59,1990	0
129054	8,8762	20,3698	-65,8883	96,7088	95,5928	65,3031	-70,1714	-4,2024	70,6785	-18,0480	1

Рисунок 1.13. Трансформований набір даних.

Отже, виконуючи послідовні трансформації текстових даних з категоріальних в числові, з текстових в кодування ембедінгами дійсних чисел, нам вдалося отримати набір повністю числових даних, які можна використовувати для машинного навчання.

1.4 Архітектури рекомендаційної системи

У розділі 1.2 було описано стратегії, які використовують рекомендаційні системи, а саме: рекомендації на основі вмісту; на основі колаборативної фільтрації; гібридні, які поєднують ці дві стратегії.

У цьому розділі буде розглянуто алгоритми колаборативної фільтрації, а саме: алгоритм memory-based і model-based системи. У якості memory-based буде розглянуто популярну модель кластеризації k-Nearest Neighbours (kNN). У якості model-based системи буде розглядатись Funk SVD і її модернізовані варіанти.

1.4.1 k-Nearest Neighbours

Популярний метод k-Nearest Neighbours використовується для вирішення широкого спектру задач: від розв’язання задачі регресії до задач класифікації, і навіть у системах рекомендацій.[14]

Цей метод базується на використанні оцінок з матриці взаємодії. Перед цим обирається напрямок, на якому буде базуватись модель, а саме: item-based або user-based. Як нам відомо з розділу 1.2.3, перший напрямок полягає в тому, щоб знаходити предмети зі схожим вектором оцінок, другий – знаходити користувачів зі схожим вектором оцінок.

Схожість векторів оцінок можна оцінити за допомогою методу скалярного добутку, евклідової відстані або косинусної відстані, опис цих методів можна знайти у розділі 1.2.1.

Після того, як було розраховано значення попарної схожості елементів, відповідно обраного напрямку, з отриманого вектора знаходять k елементів, які мають найбільше значення схожості.

Наприклад, було обрано напрямок user-based, тоді користувачу будуть рекомендувати предмети, які були високо оцінені найближчими сусідами, але не були оцінені користувачем, для якого робиться рекомендація.

Проста модель, яку можна використати, застовуючи популярну бібліотеку для машинного навчання Scikit-Learn. Але постійно розраховувати попарні оцінки схожості потребує обчислювальних потужностей, тому перейдемо до підходу model-based системи.

1.4.2 Funk SVD

Модель Funk Singular Value Decomposition (SVD) була використана Симоном Фанком на змаганнях Netflix prize з розробки кращої моделі колаборативної фільтрації, яка прогнозувала рейтинг користувача для фільмів, використовуючи лише явні відгуки (explicit feedback), а саме рейтинги, які були виставлені до цього.

Explicit feedback (явний відгук) – це прямий відгук користувача, який він надає свідомо та явно. Він включає в себе дії користувача, які чітко демонструють його вподобання та неприязнь. Наприклад, оцінки, рецензії, клацання кнопок «Подобається»/ «Не подобається» (система, яку використовує Youtube).

Implicit feedback (неявний відгук) – це непрямий зворотний зв'язок, який користувач надає без явного висловлювання своїх вподобань або неприязні. Це дані, які збираються на основі поведінки користувача. Наприклад, час проведений за переглядом фільму, переходи за посиланням та їх частота, додавання в обране.

Метод Singular Value Decomposition це метод лінійної алгебри, який використовується для розкладання (декомпозиції) матриці на три інші матриці. Це потужний інструмент, який використовується не лише у рекомендаційних системах, а і для аналізу даних, обробки зображень.

Декомпозиція матриці дозволяє відтворити матрицю R за допомогою трьох інших матриць:

$$R = U\Sigma V^*, \tag{1.5}$$

де $U -$ унітарна матриця розміру $m \times m$ над полем K, $V^* -$ унітарна матриця для матриці V розміру $n \times n$ над полем K, $\Sigma -$ діагональна матриця розміру $m \times n$ з числами σ на діагоналі, де $\sigma -$ сингулярні значення.

Використовуючи цей метод декомпозиції матриці, ми маємо діло з квадратними матрицями, які при перемножені дорівнюють рейтингу користувачів. Розмір цих матриць сягає таких масштабів, що комп'ютер не здатен завантажити їх у пам'ять.

Для вирішення цієї проблеми С. Фанк застосував до матриці рейтингів матричну факторизацію схожу на SVD, але при цьому зменшив розмірність матриць декомпозиції до числа k, яке буде відповідало за кількість важливих ознак.

Перемноження ознак буде дорівнювати рейтингам користувачів щодо конкретного фільму. Припустимо матриця ознак користувача буде $U = U\Sigma$, а матриця ознак фільмів $V = V$.

Формула моделі має наступний вигляд:

$$R \approx UV^T, \tag{1.6}$$

де $R -$ апроксимована матриця рейтингів, $U -$ матриця ознак користувачів , $V -$ матриця ознак фільмів.

Матриця R має розмір $n \times m$, де $n -$ це кількість користувачів, m $-$ це кількість фільмів. Матриця U має розмір $n \times k$, а матриця V має розмір $m \times k$, де $k -$ це кількість ознак.[10]

Формула 1.6 повторює логіку методу Singular Value Decomposition, маючи менший розмір матриць декомпозиції, але при цьому значення в результуючої матриці наближається за допомогою Stochastic Gradient Descent (SGD).

Як було вказано, вихідна матриця буде наближатись до вхідної, використовуючи SGD до матриць U та V. Щоб застосувати цей метод, треба ввести функцію втрат (loss function). Для цієї моделі функція втрат має наступний вигляд:

$$L = \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - \hat{r}_{ij}\right)^2 + \frac{\lambda}{2} \|u_i\|^2 + \frac{\lambda}{2} \left\|v_j\right\|^2, \qquad (1.7)$$

де $r_{ij} -$ рейтинг i користувача для j фільму, $\hat{r}_{ij} -$ рейтинг розрахований моделлю, $u_i -$ вектор ознак користувача i, $v_j -$ вектор ознаки фільму j.

Якщо перша частина, де розраховується різниця рейтингів відповідає за квадратичну похибку, тоді друга частина відповідає за регуляризацію L2.

Розпишемо формулу:

$$L = \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} v_{js}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2} \sum_{j=1}^{m} \sum_{s=1}^{k} v_{js}^2, \qquad (1.8)$$

де $r_{ij} -$ рейтинг i користувача для j фільму, $u_{is} -$ значення ознаки s для користувача i, $v_{js} -$ значення ознаки s для фільму j.[17]

Значення ознак будуть обиратись так, щоб мінімізувати значення функції втрат. Як можна зрозуміти, ці ознаки будуть неявними іншими словами це будуть latent features і для того, щоб з'ясувати за що відповідає кожна ознака треба провести окреме дослідження факторів.

Окрім латентних ознак ми можемо додати неявні відгуки (Implicit feedback) користувачів щодо фільмів. Тоді формула 1.4.2 може бути змінена наступним чином:

$$R \approx (U + FY)V^T, \tag{1.9}$$

де $R -$ апроксимована матриця рейтингів, $U -$ матриця ознак користувачів , $V -$ матриця ознак фільмів, $F -$ матриця неявних відгуків, $Y -$ неявні ваги для матриці неявних відгуків.[10]

Матриця F відповідає розміру матриці R, а саме $n \times m$, тобто кожне значення матриці відповідає неявному відгуку користувача до фільму. Матриця Y відповідає за перетворення матриці з розміром $n \times m$ у матрицю з розміром ознак користувача, тому вона має розмір $m \times k$.

При прогнозуванні оцінок необхідно враховувати упередження користувачів. Наприклад, користувач може поставити завищену оцінку фільму, який визнано якоюсь спільнотою. У той час як менш популярний фільм отримає помірнішу оцінку.

Або можна розділити користувачів на більш менш критичних. Хтось легко ставить високі рейтинги, хтось – навпаки. Ці упередження необхідно враховувати під час прогнозування - для цього введемо модель, яка має назву SVD++.

1.4.3 SVD++

Модель має наступну формулу:

$$\hat{r}_{ui} = \mu + b_i + b_u + p_u q_i^T, \tag{1.10}$$

де $\mu -$ середнє значення балів, виставлених по всіх пунктах всіма користувачами; $b_i -$ зміщення, яке вносить елемент i; $b_u -$ зміщення, яке вносить користувач u, $p_u -$ вектор ознак користувача, $q_i -$ вектор ознак елементу.

Значення елементів p_u, q_i, b_i, b_u розраховується під час навчання моделі, яка намагається мінімізувати наступну функцію втрат:

$$L = \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \|p_u\|^2 + \|q_i\|^2) \qquad (1.11)$$

де $\mu -$ середнє значення балів, виставлених по всіх пунктах всіма користувачами; $b_i -$ зміщення, яке вносить елемент i; $b_u -$ зміщення, яке вносить користувач u, $p_u -$ вектор ознак користувача, $q_i -$ вектор ознак елементу, $\lambda -$ коефіцієнт регуляризації.

У розділі 1.1 згадується поняття cold-start problem, під яким мається на увазі ситуація, коли невідомий користувач або фільм потрапляє у систему і вона не може визначити його ознаки до того моменту, поки не буде навчена на новому наборі даних. [17]

Потрапляння нового елементу у систему значить, що вона не знає його ознаки – це позначається тим, що вектор ознак елементу є нульовим. Наприклад, на сайт потрапляє новий користувач, вектор ознак і зміщення якого дорівнюють нулю. Якщо підставити ці значення у формулу 1.4.6 ми отримаємо наступний запис:

$$\hat{r}_{ui} = \mu + b_i \qquad (1.12)$$

Таким чином, значення формули 1.4.8 залежить лише від значення середнього рейтингу фільмів μ і його зміщення b_i. Як можна запідозрити, високі оцінки будуть отримувати фільми, які подобаються великій кількості умовних груп користувачів, наприклад, фільми, які вважаються класикою фільмографії. В той час, як фільми, направленні на конкретну групу, можуть мати низький рейтинг.

Отже, розрахунок рейтингу за допомогою моделі SVD++ для невідомого користувача дозволяє вирішити cold-start problem. Для невідомого фільму стратегія виведення його ознак схожа. Замість зміщення фільму у формулі 1.4.8 буде використано зміщення користувача. Тоді фільм буде рекомендовано для користувачів, які часто залишають високий рейтинг. Поступово ознаки фільму будуть окреслюватись по мірі збільшення кількості залишених рейтингів і донавчання моделі рекомендації.

1.5 Засоби для побудови моделі

Для побудови моделей, описаних у розділі 1.4 можна скористатись вже готовими бібліотеками, наприклад, LightFM[18]. Також можна самостійно створити модель, використовуючи, як каркас, бібліотеки для машинного навчання, наприклад, Tensorflow[11] або Pytorch[20].

TensorFlow — це потужний фреймворк для машинного навчання з відкритим вихідним кодом, розроблений компанією Google. Він забезпечує повний набір інструментів для розробки, тренування і розгортання моделей машинного навчання та глибокого навчання.

PyTorch — це фреймворк для машинного навчання з відкритим вихідним кодом, розроблений Facebook's AI Research lab (FAIR). Він широко використовується як у дослідницьких, так і у промислових проектах завдяки своїй простоті у використанні та потужним можливостям.

Підхід з використанням фреймворків для машинного навчання дозволить розробнику керувати структурою моделі як йому заманеться, тому що розробник сам вибирає архітектуру, розробляє алгоритми та функції, функцію оптимізації, при цьому може також використовувати вбудовані функції з цих бібліотек.

Великою перевагою використання таких фреймворків є те, що розробивши клас моделі, розробник може навчити її, не розробляючи для

цього функцію пошуку градієнту, тому що це може зробити сама модель, яка успадкувала цю функцію зі стандартного класу.

1.6. Висновки до розділу

У цьому розділі було розглянуто загальні відомості про рекомендаційні системи у середовищі, де присутній покупець і товар, який в тій чи іншій мірі йому подобається. Розробка рекомендаційної системи може бути виконана за різними стратегіями: або створити модель, яка здатна одночасно розрахувати рейтингову оцінку для кожної пари елементів; або виконувати для кожного користувача певний алгоритм, який буде рекомендувати нові пропозиції, спираючись на міру схожості, виконуючи кожен раз велику кількість попарних оцінок об’єктів системи, з ростом яких будуть збільшуватись потреби в обчислювальних потужностях. Для збільшення ефективності також застосовують гібридну стратегію, яка поєднує обидва варіанти.

У цій роботі було обрано стратегію побудувати модель, яка в результаті дасть матриці ознак елементів. Так як розрахунок виконується спираючись лише на оцінки взаємодії, матриці ознак будуть формуватися за принципом латентних ознак, де зміст кожної ознаки є скритим, тому що розрахунок виконувався стохастичним градієнтним спуском. Для пошуку змісту ознак потрібно виконувати окремі дослідження, які можуть бути неефективними через те, що при кожній новій побудові моделі зміст ознак буде змінюватись. Щоб уникнути цих додаткових досліджень, можна ввести в модель вже знайомі ознак, а саме матрицю ознак фільмів – таким чином, в залежності від архітектури моделі, ознаки фільмів перейдуть в ознаки користувача.

Щоб перетворити інформацію про фільм у вигляд, який зможе опрацювати комп’ютер, може бути застосовано метод One-hot-encoding, який перетворить категоріальну змінну у числову шляхом створення для кожної

категорії окремий стовпчик. Також ця стратегія може бути застосована до року випуску, який поділено на десятиріччя – таким чином ми даємо змогу врахувати відношення користувача до окремої епохи. Кодування можна застосувати не лише до категоріальних даних, а і до текстових, які описують об’єкти у вигляді коротких речень. Для того, щоб зробити це, треба скористатись методами обробки природньої мови, а саме виконати кодування тексту, але використовуючи більш продвинутий варіант, ніж one-hot-encoding. Замість вектору бінарних значень можна використати ембединги словника, який побудовано за методом GloVe. Цей вид кодування має перевагу в тому, що дозволяє виконувати операцію додавання для ембедингів слів, при цьому зберігати їх семантичний зв’язок.

Після підготовки даних можна перейти до побудови і навчання моделей, які базуються на матричній факторизації. У ці моделі можна внести готову матрицю ознак і зафіксувати її, щоб виконувати мінімізацію функції втрат за рахунок інших елементів моделі, а саме: векторів зміщень, інших матриць ознак тощо.

Як було вказано раніше, внесення матриці ознак може дозволити перенести зміст однієї матриці на іншу, але це явище може бути нівельовано великою кількістю елементів, які змінюються стохастичним градієнтним спуском. Наприклад, модель SVD++ має, окрім матриць ознак, константу у вигляді середньої рейтингової оцінки, а також вектори зміщень для кожного об’єкту будь то користувач або фільм. Тому в такій моделі досягти ефект переносу ознак буде важче, ніж в моделі Funk SVD, яка використовує лише перемноження матриць ознак. Не дивлячись на цей недолік, SVD++ має таку архітектуру, що дозволяє використовувати її для рішення проблеми холодного старту, яка полягає в тому, що об’єкт, для якого треба виконати рекомендацію, ще не має своїх ознак. Цей ефект було досягнуто за рахунок використання середньої рейтингової оцінки і використання векторів зміщень об’єктів.

Для самостійної побудови перелічених моделей можна скористатись бібліотеками Tensorflow або Pytorch, які дозволяють розробити модель з власною архітектурою, при цьому не витрачати час на написання методів, які обслуговують модель, наприклад, метод градієнтного спуску, прогнозування тощо. Такий підхід дозволить розробнику сконцентрувати всю увагу на розробці логіки моделі.

2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Опис об’єкту дослідження

Об’єктом дослідження є база відгуків користувачів сайту MovieLens.[13] Відгуки містять в собі рейтингову оцінку користувача для фільму, а також користувач може додати теги, в яких він зазначає тему, яка висвітлена у фільмі: місце, де відбуваються події; акторський склад; провідна тема; жанр тощо.

Хоча дані у базі можуть частково збігатися з інформацією, представленою на IMDb, MovieLens не імпортує свої дані безпосередньо з IMDb. Натомість вони покладаються на активність своїх власних користувачів для створення унікального набору даних.

IMDb (Internet Movie Database) — це один із найпопулярніших у світі веб-сайтів, що пропонує огляди, рейтинги та інформацію про фільми, телепередачи та зірок кіно.

Остання база даних, яку вивантажував у мережу сайт MovieLens, налічує 25 мільйонів записів рейтингів з датою, проте у цій кваліфікаційній роботі було обрано записи з дати «01/01/2019» по «20/07/2023» і сумарна кількість записів дорівнює 1 077 440. Кількість унікальних користувачів у вибраному проміжку часу дорівнює 2500, а фільмів – 1557.

Загальний архів з даними дає доступ до переліку фільмів, які містять унікальний Id і Id для сайту IMBD, а також мають поле з жанрами, які притаманні фільму. Також архів дає доступ до тегів, які залишають користувачі до конкретного фільму. Якість цих тегів можна оцінити за допомогою таблиці відповідності конкретного тегу до конкретного фільму. Це значить, що не всі теги, які користувачі залишили у відгуку, зможуть бути використані, тому що неможливо визначити їх відповідність.

2.2 Структура моделі даних

Перед початком роботи з даними, які пропонує сайт, ознайомимся з їх структурою і особливостями. Кожна таблиця зберігає дані у відповідному форматі і стосується однієї частини загальної інформації. Наприклад, інформація про рейтинги зберігається в одній таблиці, а теги до цих рейтингів в іншій. Інформацію стосовно кожного запису можна поєднати за допомогою кортежу ключів ідентифікаторів фільму і користувача.

Опишемо таблиці, які зберігають інформацію, та розглянемо декілька записів з них.

Таблиця reviews (відгуків, рейтингів) складається з наступним полів:

1. Унікальний ідентифікатор особи;
2. Унікальний ідентифікатор фільму;
3. Рейтинг;
4. Дата у форматі Unix;

Приклади записів з таблиці зображено на рисунку 2.1.

	userId	movieId	rating	timestamp
1	13	1025	5.0	1593485253
2	13	1197	5.0	1593485512
3	13	2033	5.0	1593485225
4	13	2139	5.0	1593485228
5	13	2140	5.0	1593485408
6	13	2161	5.0	1593485397
7	13	2193	5.0	1593485422
8	13	2690	5.0	1593484577
9	13	2761	5.0	1593485243
10	13	4306	5.0	1593485379
11	13	27706	2.0	1593485314
12	13	50601	4.5	1593485425

Рисунок 2.1. Приклад записів з таблиці відгуків.

Наступна таблиця movies (фільми) складається з наступним полів:

1. Унікальний ідентифікатор фільму;

2. Назва;
3. Жанри;

Приклади записів з таблиці зображено на рисунку 2.2.

	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller
11	11	American President, The (1995)	Comedy|Drama|Romance
12	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
13	13	Balto (1995)	Adventure|Animation|Children
14	14	Nixon (1995)	Drama
15	15	Cutthroat Island (1995)	Action|Adventure|Romance

Рисунок 2.2. Приклад записів з таблиці фільмів.

Наступна таблиця tags (теги) складається з наступним полів:

1. Унікальний ідентифікатор особи;
2. Унікальний ідентифікатор фільму;
3. Тег;
4. Дата у форматі Unix;

Приклади записів з таблиці зображено на рисунку 2.3.

	userId	movieId	tag	timestamp
1	37	47	Kevin Spacey	1578167238
2	37	47	Morgan Freeman	1578167242
3	37	47	powerful ending	1578167246
4	37	47	twist ending	1578167240
5	37	165	action	1578164108
6	37	293	Gary Oldman	1578165432
7	37	293	great acting	1578165426
8	37	293	Jean Reno	1578165421
9	37	293	Natalie Portman	1578165423
10	37	480	classic	1578166087
11	37	480	Steven Spielberg	1578166078
12	37	527	acting	1578163193
13	37	527	John Williams	1578163155
14	37	527	moving	1578163178
15	37	541	classic	1578164881
16	37	541	dystopia	1578164862

Рисунок 2.3. Приклад записів з таблиці тегів.

Наступна таблиця links (посилання) складається з наступним полів:

1. Унікальний ідентифікатор фільму;
2. Унікальний ідентифікатор фільму у форматі IMBD;
3. Унікальний ідентифікатор фільму у форматі TMBD;

Приклади записів з таблиці зображено на рисунку 2.4.

	movieId	imdbId	tmdbId
1	1	114709	862
2	2	113497	8844
3	3	113228	15602
4	4	114885	31357
5	5	113041	11862
6	6	113277	949
7	7	114319	11860
8	8	112302	45325
9	9	114576	9091
10	10	113189	710
11	11	112346	9087
12	12	112896	12110
13	13	112453	21032
14	14	113987	10858
15	15	112760	1408
16	16	112641	524
17	17	114388	4584
18	18	113101	5
19	19	112281	9273
20	20	113845	11517

Рисунок 2.4. Приклад записів з таблиці посилань.

Наступна таблиця genome-tags (набір тегів) складається з наступним полів:

1. Унікальний ідентифікатор тегу;
2. Тег;

Приклади записів з таблиці зображено на рисунку 2.5.

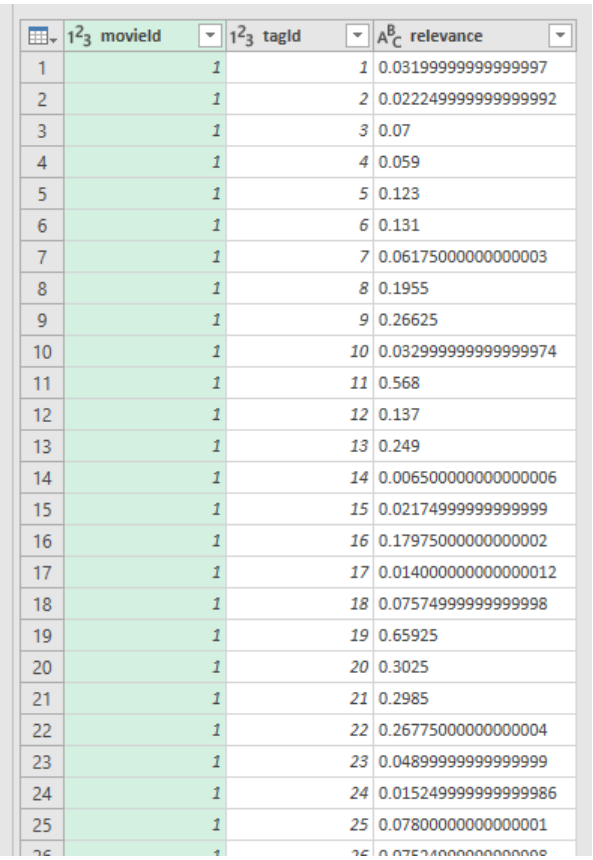
	tagId	tag
1	1	007
2	2	007 (series)
3	3	18th century
4	4	1920s
5	5	1930s
6	6	1950s
7	7	1960s
8	8	1970s
9	9	1980s
10	10	19th century
11	11	3d
12	12	70mm
13	13	80s
14	14	9/11
15	15	aardman
16	16	aardman studios
17	17	abortion
18	18	absurd
19	19	action
20	20	action packed
21	21	adaptation

Рисунок 2.5. Приклад записів з таблиці посилань.

Наступна таблиця genome-scores (оцінка тегів) складається з наступним полів:

1. Унікальний ідентифікатор фільму;
2. Унікальний ідентифікатор тегу;
3. Відповідність тегу для фільму;

Приклади записів з таблиці зображено на рисунку 2.6.



	movieId	tagId	relevance
1	1	1	0.03199999999999997
2	1	2	0.022249999999999992
3	1	3	0.07
4	1	4	0.059
5	1	5	0.123
6	1	6	0.131
7	1	7	0.06175000000000003
8	1	8	0.1955
9	1	9	0.26625
10	1	10	0.032999999999999974
11	1	11	0.568
12	1	12	0.137
13	1	13	0.249
14	1	14	0.006500000000000006
15	1	15	0.02174999999999999
16	1	16	0.17975000000000002
17	1	17	0.014000000000000012
18	1	18	0.07574999999999998
19	1	19	0.65925
20	1	20	0.3025
21	1	21	0.2985
22	1	22	0.26775000000000004
23	1	23	0.04899999999999999
24	1	24	0.015249999999999986
25	1	25	0.07800000000000001

Рисунок 2.6. Приклад записів з таблиці відповідності тегів для фільмів.

2.3 Попередня обробка даних

Представлений набір даних був скомпонований з метою максимально зменшити пам’ять. Наприклад, розріджена матриця взаємодії користувача з елементами, яка може мати нульові значення як більшу частину усіх значень, була записана у вигляді таблиці (таблиця відгуків, див. рисунок 2.2), де указано для якого рядка і стовпчика матриці значення буде ненульове.

Для того, щоб прогнозувати рейтинги взаємодії у вигляді матриці треба перевести значення з табличного виду у матричний, прибравши стовпчики з ідентифікаторами. На рисунку 2.7 зображено фрагмент початкового вигляду даних про відгук.

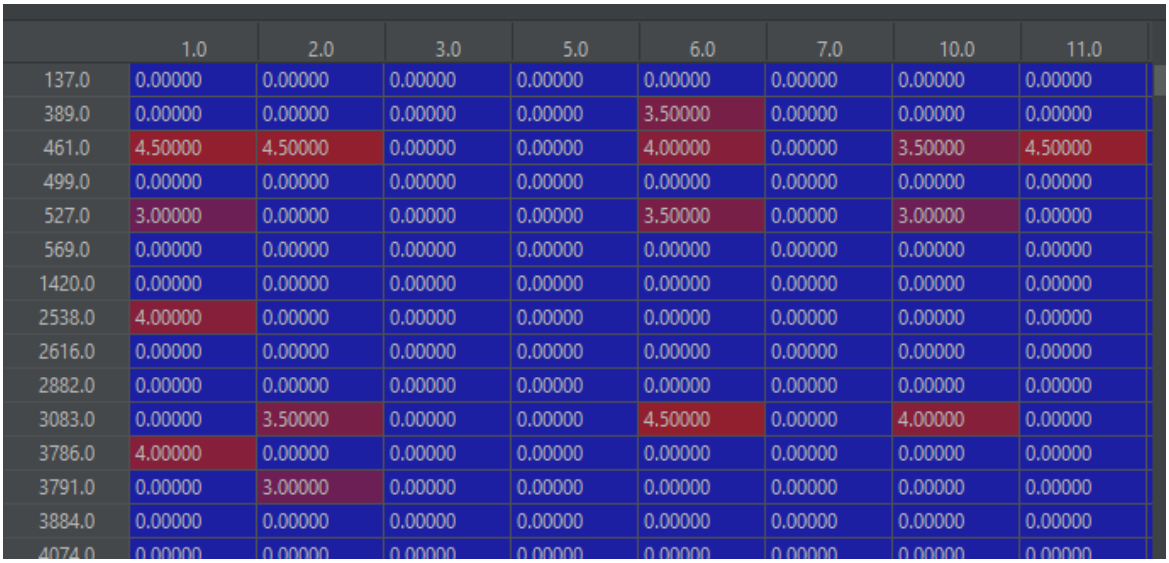
userId	movieId	rating	timestamp
137.00000	1394.00000	4.00000	1668055552.00000
137.00000	7147.00000	4.00000	1674365952.00000
137.00000	50872.00000	3.50000	1680910720.00000
137.00000	96610.00000	4.50000	1646890752.00000
137.00000	101864.00000	3.50000	1669579648.00000
137.00000	134853.00000	5.00000	1670644224.00000
389.00000	481.00000	4.00000	1643258624.00000
389.00000	535.00000	3.50000	1645830656.00000
389.00000	628.00000	4.00000	1644352512.00000
389.00000	784.00000	3.50000	1657310208.00000
389.00000	908.00000	3.50000	1673913216.00000
389.00000	1178.00000	4.50000	1650321280.00000
389.00000	1208.00000	3.00000	1669248640.00000
389.00000	1237.00000	3.50000	1675809280.00000
389.00000	1250.00000	4.00000	1644974848.00000
389.00000	1252.00000	4.00000	1667505408.00000

Рисунок 2.7. Початковий вигляд таблиці відгуків.

Кількість унікальних користувачів і фільмів у неопрацьованому вигляді не дозволяє створити розріджену матрицю і загрузити її у пам’ять для подальшої обробки, тому для цього обмежимо записи у таблиці датами. Так як було вказано раніше, останній запис датується «20/07/2023», тоді можна обрати початкову дату, наприклад, «01/01/2019» – значить буде обрано дані за 4,5 роки. Лістинг програмного коду для цього завдання можна знайти у додатку Г.

Після обмеження даних за допомогою дат, було отримано набір з 4315 унікальних користувачів і 1557 унікальних фільмів. Для створення навчального і тестового набору було виконано пересікання множини унікальних користувачів і фільмів. Після цього набір даних було ще раз урізано, за допомогою вибору користувачів, які залишили більше 50 відгуків – таким чином ми вибрали множину користувачів, на яких зможемо навчити модель уособлювати інтереси. Тепер набор містить 1214 користувачів на 1557 фільмів.

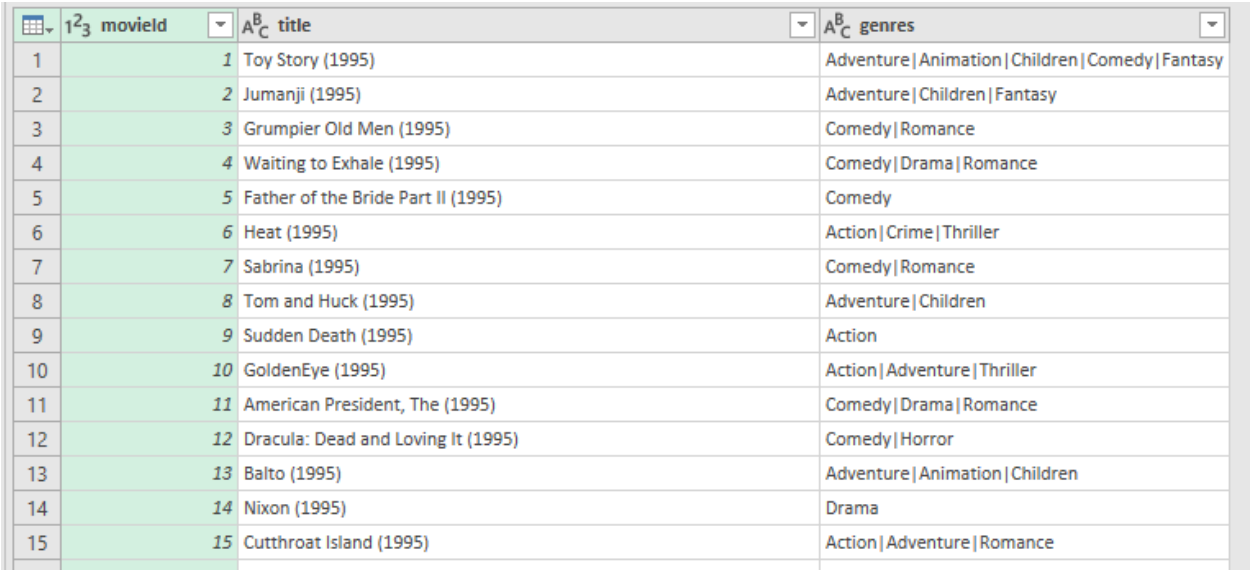
На рисунку 2.8 зображено фрагмент вже «обрізаної» таблиці відгуків, яка трансформована у розріджену матрицю, де рядок відповідає за конкретного користувача, а стовпчик за фільм.



	1.0	2.0	3.0	5.0	6.0	7.0	10.0	11.0
137.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
389.0	0.00000	0.00000	0.00000	0.00000	3.50000	0.00000	0.00000	0.00000
461.0	4.50000	4.50000	0.00000	0.00000	4.00000	0.00000	3.50000	4.50000
499.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
527.0	3.00000	0.00000	0.00000	0.00000	3.50000	0.00000	3.00000	0.00000
569.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1420.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2538.0	4.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2616.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2882.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3083.0	0.00000	3.50000	0.00000	0.00000	4.50000	0.00000	4.00000	0.00000
3786.0	4.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3791.0	0.00000	3.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3884.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4074.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Рисунок 2.8. Фрагмент розрідженої матриця відгуків.

Наступним для обробки буде таблиця з фільмами, яка містить в собі ідентифікатор фільму, назву з роком випуску, жанри. На рисунку 2.9 можна побачити фрагмент цієї таблиці.



	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller
11	11	American President, The (1995)	Comedy|Drama|Romance
12	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
13	13	Balto (1995)	Adventure|Animation|Children
14	14	Nixon (1995)	Drama
15	15	Cutthroat Island (1995)	Action|Adventure|Romance

Рисунок 2.9. Фрагмент таблиці фільмів.

Як можна побачити, більшість даних мають текстовий тип, який не підходить для обробки комп'ютером під час машинного навчання. Для початку трансформуємо стовпчик жанрів за допомогою one-hot-encoding, який було описано у розділі 1.3.1. Для кожного жанру буде створено окремий стовпчик, в якому записано 0, якщо жанр не притаманний фільму, 1 - якщо навпаки. Рік випуску фільму буде винесено в окремий стовпчик, який буде поділено на десятиріччя і трансформовано за допомогою one-hot-encoding на 6 стовпчиків: (age_1950-70s, age_1980s, age_1990s, age_2000s,age_2010s, age_older1950s).

Трансформована таблиця буде мати наступний вигляд (див. рисунок 2.10, 2.11).

movieId	title	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror
2	Jumanji	0	0	1	0	1	0	0	0	0	1	0	0
3	Grumpier Old Men	0	0	0	0	0	1	0	0	0	0	0	0
4	Waiting to Exhale	0	0	0	0	0	1	0	0	1	0	0	0
5	Father of the Bride Part II	0	0	0	0	0	1	0	0	0	0	0	0
6	Heat	0	1	0	0	0	0	1	0	0	0	0	0
7	Sabrina	0	0	0	0	0	1	0	0	0	0	0	0
8	Tom and Huck	0	0	1	0	1	0	0	0	0	0	0	0
9	Sudden Death	0	1	0	0	0	0	0	0	0	0	0	0
10	GoldenEye	0	1	1	0	0	0	0	0	0	0	0	0
11	American President, The	0	0	0	0	0	1	0	0	1	0	0	0
12	Dracula: Dead and Loving It	0	0	0	0	0	1	0	0	0	0	0	1
13	Balto	0	0	1	1	1	0	0	0	0	0	0	0
14	Nixon	0	0	0	0	0	0	0	0	1	0	0	0
15	Cutthroat Island	0	1	1	0	0	0	0	0	0	0	0	0
16	Casino	0	0	0	0	0	0	1	0	1	0	0	0
17	Sense and Sensibility	0	0	0	0	0	0	0	0	1	0	0	0
18	Four Rooms	0	0	0	0	0	1	0	0	0	0	0	0
19	Ace Ventura: When Nature Calls	0	0	0	0	0	1	0	0	0	0	0	0
20	Money Train	0	1	0	0	0	1	1	0	1	0	0	0
21	Get Shorty	0	0	0	0	0	1	1	0	0	0	0	0

Рисунок 2.10. Фрагмент трансформованої таблиці фільмів.

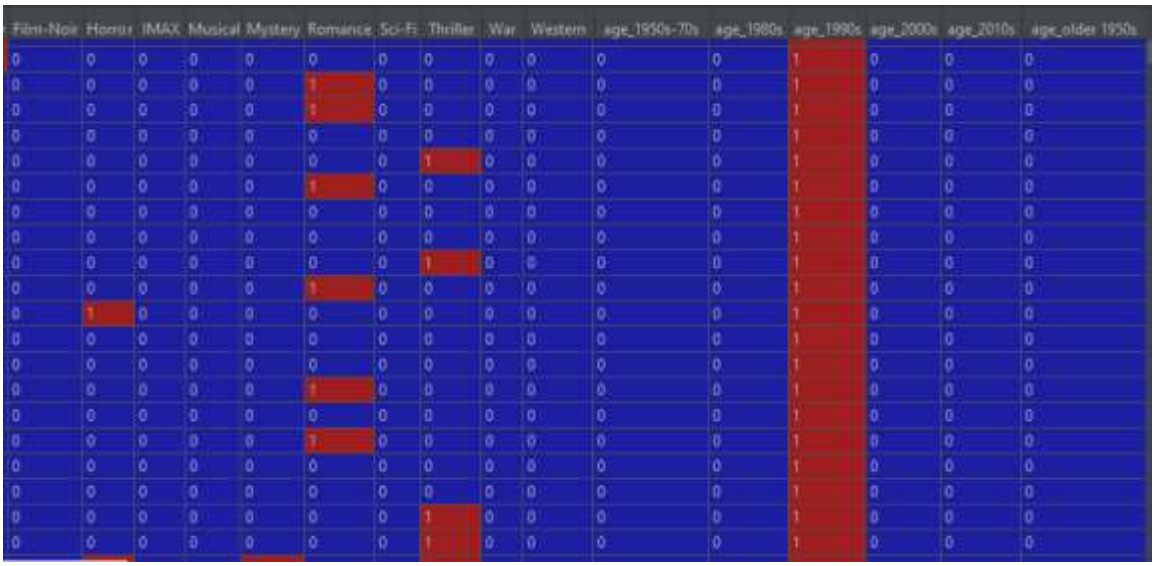


Рисунок 2.11 Фрагмент трансформованої таблиці фільмів.

Отримана таблиця тепер може бути використана, як сховище ознак фільмів, з якого ми можемо діставати ознаки фільму за допомогою його ідентифікатора.

Перейдемо до таблиці з тегами користувачів, які вони могли залишити разом з рейтинговою оцінкою фільму. З цього можна зрозуміти, що не кожен користувач залишив теги. Також, для того, щоб оцінити якість цих тегів, розробник набору даних залишив таблицю відповідності тегів до фільму. Тому для коректної роботи треба перевірити, щоб кожен залишений тег мав значення відповідності до фільму.

На рисунку 2.3 зображено теги користувачів, але вони не мають ідентифікатора, як на рисунку 2.5 або 2.6. Для того, щоб ефективно використовувати ці теги, ми повинні знайти їх ідентифікатори. Кількість усіх записів у таблиці з тегами користувачів дорівнює 2 328 315, але після знаходження їх ідентифікаторів залишилось всього 1 019 936 записів. Пошук ідентифікаторів тегів відбувався за допомогою порівняння текстового поля «tag». Перед цим усі текстові значення було форматовано до регістру малих літер. Лістинг програмного коду для цього завдання можна знайти у додатку Е. Кінцевий вигляд опрацьованих тегів можна побачити на рисунку 2.12.

userId	movieId	tag	timestamp	tagId	relevance
10	260	sci-fi	1430666538	887.00000	0.95250
14	1221	mafia	1311600746	622.00000	0.97925
14	58559	atmospheric	1311530439	86.00000	0.65725
14	58559	batman	1311530391	117.00000	0.98125
14	58559	comic book	1311530398	232.00000	0.91600
14	58559	dark	1311530428	285.00000	0.95475
14	58559	imdb top 250	1311530451	536.00000	0.97750
14	58559	oscar (best supporting actor)	1311530432	758.00000	0.97925
14	58559	psychology	1311530417	824.00000	0.68650
14	58559	superhero	1311530388	989.00000	0.97900
14	58559	vigilante	1311530423	1082.00000	0.98775
14	58559	violence	1311530444	1084.00000	0.83900
16	57183	family	1491353765	374.00000	0.31525
26	296	crime	1429398919	268.00000	0.94025
26	296	cult film	1429398919	276.00000	0.79700
37	47	powerful ending	1578167246	806.00000	0.97675
37	47	twist ending	1578167240	1050.00000	0.95900

Рисунок 2.12. Таблиця тегів після обробки.

Як можна побачити, не всі теги відповідають тим тегам, які насправді притаманні фільму. Наприклад, користувач з ідентифікатором 16 залишив тег «family», але релевантність тегу відносно фільму, до якого було залишено відгук, всього 0.31 на інтервалі (0;1).

2.4. Розвідувальний аналіз

Після попередньої обробки даних виконаємо розвідувальний аналіз - почнемо з оцінок, а саме розподілу їх значень(див. рисунок 2.13).

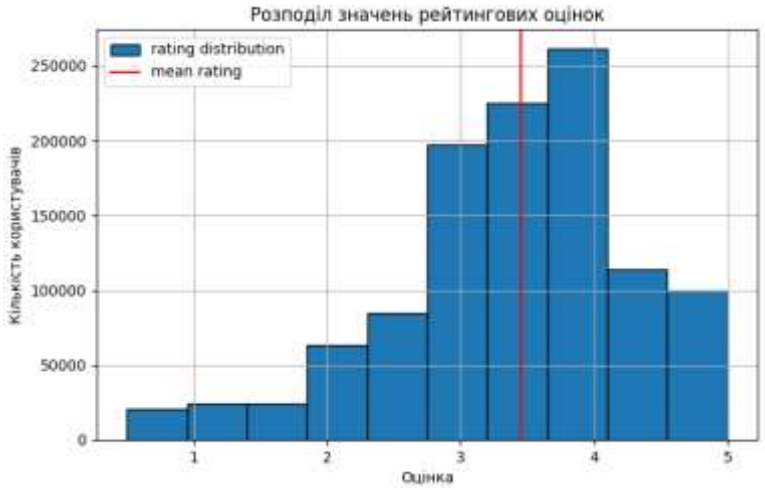


Рисунок 2.13. Графік розподілу значень рейтингових оцінок.

Як можна побачити, рейтингові оцінки вимірюються за допомогою 5 бальної системи, а розподіл оцінок скошений в сторону високого рейтингу, червоною вертикальною лінією виділено середній рейтинг, а саме приблизно 3.5.

Далі розглянемо список фільмів, які були найчастіше оцінені і ті, що мають найвищу середню оцінку. Можна помітити, що популярний фільм, який має велику кількість оцінок, буде мати великий середній рейтинг (див. рисунок 2.14).

movieId	count	title	mean_rating
207313.00000	2711	Knives Out	3.89082
204698.00000	2686	Joker	3.78388
202439.00000	2681	Parasite	4.16374
122914.00000	2670	Avengers: Infinity War - Part II	3.70056
195159.00000	2315	Spider-Man: Into the Spider-Verse	4.04039
202429.00000	2283	Once Upon a Time in Hollywood	3.66929
122912.00000	2052	Avengers: Infinity War - Part I	3.66277
79132.00000	1928	Inception	4.00752
164179.00000	1927	Arrival	3.92605
109487.00000	1894	Interstellar	3.90180
2571.00000	1884	Matrix, The	4.16587
58559.00000	1878	Dark Knight, The	4.09984
201773.00000	1841	Spider-Man: Far from Home	3.48180
112852.00000	1831	Guardians of the Galaxy	3.71491
122906.00000	1813	Black Panther	3.34666
122904.00000	1806	Deadpool	3.60936
115149.00000	1796	John Wick	3.69237
4993.00000	1794	Lord of the Rings: The Fellowship of the Ring, The	4.03122
122916.00000	1792	Thor: Ragnarok	3.67662
68157.00000	1791	Inglourious Basterds	3.91011
205156.00000	1790	Jojo Rabbit	3.81453

Рисунок 2.14. Фільми відсортовані за кількістю рейтингів.

movieId	count	title	mean_rating
1203.00000	1177	12 Angry Men	4.24087
318.00000	1733	Shawshank Redemption, The	4.22591
296.00000	1782	Pulp Fiction	4.18883
858.00000	1595	Godfather, The	4.17335
2571.00000	1884	Matrix, The	4.16587
202439.00000	2681	Parasite	4.16374
2959.00000	1780	Fight Club	4.15758
1221.00000	1301	Godfather: Part II, The	4.12913
2019.00000	728	Seven Samurai (Shichinin no samurai)	4.10027
58559.00000	1878	Dark Knight, The	4.09984
1214.00000	1583	Alien	4.09792
5618.00000	1470	Spirited Away (Sen to Chihiro no kamikakushi)	4.08946
593.00000	1731	Silence of the Lambs, The	4.08377
1213.00000	1415	Goodfellas	4.07951
1201.00000	1038	Good, the Bad and the Ugly, The (Buono, il brutto,...	4.06936
589.00000	1559	Terminator 2: Judgment Day	4.06575
904.00000	1029	Rear Window	4.05928
47.00000	1689	Seven (a.k.a. Se7en)	4.05417
1196.00000	1664	Star Wars: Episode V - The Empire Strikes Back	4.04808
3000.00000	1030	Princess Mononoke (Mononoke-hime)	4.04515

Рисунок 2.15. Фільми відсортовані за середній рейтингом.

Перейдемо до ознак фільмів, а саме розглянемо який жанр частіше потрапляється у наборі даних. На рисунку 2.16 зображено гістограму з кількістю появ кожного жанру.

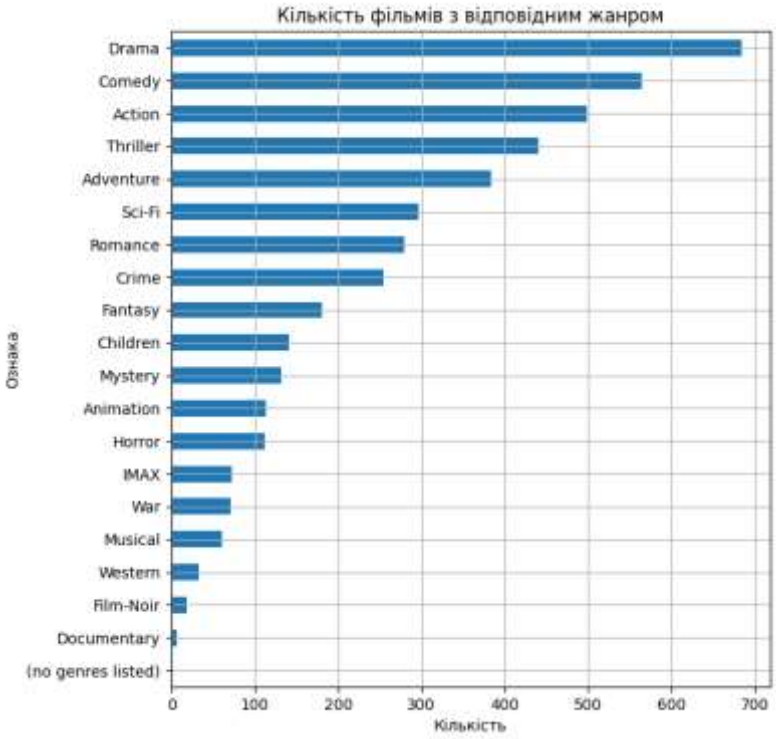


Рисунок 2.16. Кількість фільмів з відповідним жанром.

Проаналізувавши рисунок 2.16 можна зробити висновок, що жанри драма, комедія і екшн потрапляються частіше за інші у нашому наборі даних.

Далі розглянемо розподіл фільмів за відрізками випуску, а саме дізнаємося фільми якого десятиріччя більше представленні у наборі даних (див. рисунок 2.17).

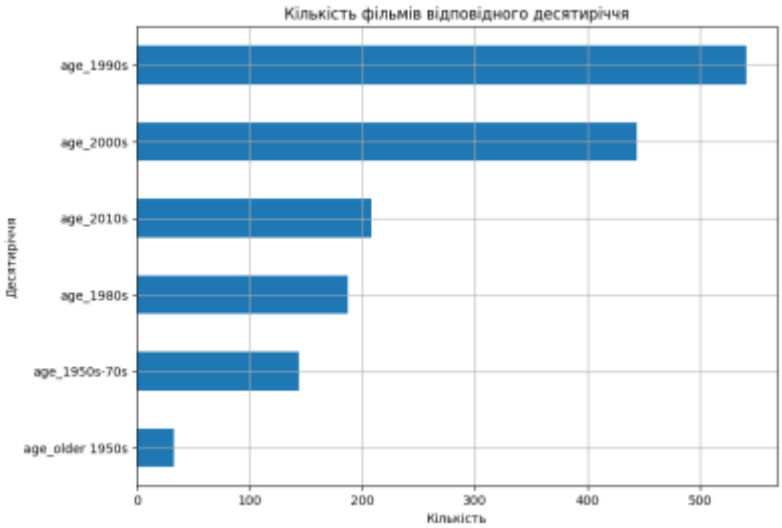


Рисунок 2.17 Розподіл фільмів за десятиріччями.

Як можна побачити на рисунку 2.17, фільми 1990х та 2000х найбільше представлені у наборі даних.

Дослідимо теги для фільмів, а саме серед мільйона відгуків було зафіксовано 4796 унікальних користувачів, серед яких всього 2107 залишили тег у відгуку. Розглянемо список найпопулярніших тегів, які користувачі залишали у відгуках (див. рисунок 2.18).

tagId	count	tag
887.00000	14892	sci-fi
86.00000	12776	atmospheric
19.00000	11747	action
230.00000	10966	comedy
417.00000	9851	funny
995.00000	9443	surreal
1091.00000	9086	visually appealing
1050.00000	8428	twist ending
286.00000	7777	dark comedy
1024.00000	7731	thought-provoking
107.00000	7718	based on a book
863.00000	7123	romance
336.00000	7056	dystopia
212.00000	6645	cinematography
377.00000	6469	fantasy
936.00000	6425	social commentary
1084.00000	6377	violence
982.00000	6201	stylized

Рисунок 2.18. Теги, які частіше всього використовували користувачі.

Як можна побачити, тег sci-fi (наукова фантастика) найбільше представлений у тегах користувачів. Якщо звернутись до списку найпопулярніших фільмів (див. рисунок 2.14), то можна помітити представників, які містять ознаки sci-fi, а саме: «Inception», «Arrival», «Interstellar», «The Matrix», «Guardians of the Galaxy», «Black Panther», «Spider-Man: Into the Spider-Verse».

2.5 Побудова моделей рекомендації

Після попередньої обробки даних і розвідувального аналізу, ми можемо приступити до побудови моделей рекомендації за методиками описаними у інформаційно-аналітичному розділі. У цій роботі будуть порівнюватись моделі матричної факторизації, які будуть наближати рейтингову оцінку користувача до тієї, яку він виставив. Це наближення буде відбуватись за допомогою стохастичного градієнтного спуску, тобто модель буде використовувати машинне навчання – це значить, що треба створити як мінімум два набори даних: навчальний і тренувальний. За допомогою

першого набору даних модель буде вчитися узагальнювати знання, а за допомогою другого набору – перевіряти рівень узагальнення цих знань.

Моделі, які будуть розглянуті у наступних розділах були побудовані на базі бібліотеки Tensorflow версії 2.10 з використанням плагіна DirectML, який дозволяє виконувати обчислення на графічному прискорювачі. Бібліотека дозволяє створити модель з нуля, використовуючи примітивні класи, від яких успадковуються функції та методи. Такий підхід робить нас незалежним від бібліотек з готовим моделями і закритим функціоналом. Розробник може створити модель з власною логікою, при цьому не відволікатись на реалізацію функцій, які обслуговують цю модель, наприклад, функція пошуку градієнту. Таким чином, розробник може сконцентрувати всю свою увагу на логіку і архітектуру моделі. Програмний код створення класу можна знайти у додатку В.

2.5.1 Набір даних для навчання і тестування моделей

Для того, щоб наблизити умови роботи моделі до реальних, уявімо ситуацію, що ми маємо інформацію про вподобання користувачів за останні три роки і нам потрібно визначити, який товар порекомендувати йому у майбутньому році. Звідси виходить, що навчальний набір даних буде містити інформацію про відгуки з «01/01/2019» по «01/01/2022», а тестовий – з «01/01/2022» по «20/07/2023» (ця дата є останньою, яка присутня у наборі даних).

Ще одним нюансом до побудови моделей є те, що ми маємо достатньо інформації про фільм, а саме: жанр, назву, рік випуску, притаманні теги. В той час як про користувача нам відома лише оцінка і його теги. У розділі 1.3.7 згадувалось поняття latent features, яка має на увазі те, що під час наближення матриці взаємодії методом стохастичного градієнтного спуску, ми не знаємо якими саме ознаками описуємо користувачів і фільм. Тому додатковим завданням для рекомендаційної системи буде не лише виконати правильний прогноз, а також розрахувати осмислені ознаки користувача. Ця

логіка виходить з того, що ознаки фільмів записані у бінарній формі, тому, за допомогою ознак користувача, ми можемо коригувати вплив окремої ознак на рейтингову оцінку, як результат, ми отримаємо її коефіцієнт важливості (вподобання) для користувача.

Отже, після виділення навчального і тестового наборів даних за допомогою граничних дат і обмеження по кількості відгуків, було сформовано матрицю взаємодії розміром 1214 користувачів на 1557 фільмів.

Як було вказано, факторизація матриці за допомогою Funk SVD дозволяє зменшити розмір матриць декомпозиції з квадратних (наприклад, 1214 на 1214) до 1214 на k, де k – це кількість ознак. У нашому випадку кількість ознак диктується матрицею ознак фільмів, яка має наступний вигляд:

Рисунок 2.19 Матриця ознак фільмів.

Рисунок 2.20. Матриця ознак фільмів.

Як можна побачити, з матриці зникли стовпчики з назвою фільму і ідентифікатором. Загальна кількість фільмів дорівнює 1557, а кількість стовпчиків 25. Отже, матриця ознак користувача буде мати розмір 1214 на 25. В основному ми будемо використовувати ці 3 матриці (матриці взаємодії, ознак користувачів, ознак фільмів), але можуть застосовуватись додаткові матриці для окремих моделей.

2.5.2 Метрика якості моделі

Для того, щоб навчити модель потрібно ввести функцію втрат, яка буде відрізнятись між моделями, тому також буде введено додаткові метрики, які урівняють всі моделі між собою і відобразить їх продуктивність. У якості таких метрик буде використано *Root Mean Squared Error* і $precision@k$.

Root Mean Squared Error (RMSE) це один з найпопулярніших показників для оцінки якості моделей регресії. Він вимірює середню різницю між передбаченими значеннями моделі та фактичними значеннями. Наведемо формулу цієї метрики:

Метрика $precision@k$ вимірює частку правильно ранжованих елементів серед перших k результатів. Вона дозволяє оцінити, наскільки модель

успішно ідентифікує релевантні елементи серед топ k , що є критично важливим для рекомендаційних моделей. Формула цієї метрики наступна:

$$RMSE = \frac{1}{n} \sum_{i,j \in Train}^{S} \left(r_{ij} - \hat{r}_{ij}\right)^2 , \quad (2.1)$$

де r_{ui} −справжнє значення рейтингу, $\hat{r}_{ui}$ − прогнозоване значення рейтингу.

Метрика $precision@k$ вимірює частку правильно ранжованих елементів серед перших k результатів. Вона дозволяє оцінити, наскільки модель успішно ідентифікує релевантні елементи серед топ k , що є критично важливим для рекомендаційних моделей. Формула цієї метрики наступна:

$$precision@k = \frac{true\ positive@k}{k} \quad (2.2)$$

Для того, щоб визначити чи є елемент позитивним, розділимо елементи матриці взаємодії на нульові і ненульові. Якщо елемент є ненульовим, але з низьким рейтингом, це значить, що користувач проявив інтерес до цього елементу, хоч він йому і не сподобався в результаті. Для виділених ненульових елементів розраховується наступне значення:

$$similarity_{ui} = cosine\ similarity(u, i) + \frac{r_{ui}}{5} \ , \quad (2.3)$$

де r_{ui} − значення рейтингу

Формула розраховує косинусну схожість ознак користувача і елементу, яка має значення на проміжку [-1;1], до якої додається відносна оцінка рейтингу. Ненульові елементи сортируються за значенням relevance і потім обирається перші k елементів, які вважаються ground truth елементами, тобто за цим списком будуть визначати true positive рекомендації.

2.5.3 Funk SVD

Перша модель, яку буде розглянуто – це Funk SVD (див. розділ 1.4.2) , яка використовує лише матриці ознак. Виконаємо навчання моделі за допомогою розробленого класу моделі на базі бібліотеки Tensorflow. Листинг

програмного коду можна знайти у додатку Г. Рейтингова оцінка буде розраховуватись за наступною формулою:

$$r_{ij} = u_i * v_j = \sum_{s=1}^{k} u_{is} v_{js} \ , \tag{2.4}$$

де r_{ij} – рейтинг i користувача для j фільму, $\hat{r}_{ij}$ – рейтинг розрахований моделлю, u_i – вектор ознак користувача i, v_j – вектор ознаки фільму j.

Функція витрат для цієї моделі буде наступна:

$$L = \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - \hat{r}_{ij}\right)^2 + \frac{\lambda}{2} \|u_i\|^2 + \frac{\lambda}{2} \left\|v_j\right\|^2 \tag{2.5}$$

де r_{ij} – рейтинг i користувача для j фільму, u_{is} – значення ознаки s для користувача i, v_{js} – значення ознаки s для фільму j.[17]

Коефіцієнт регулярізації λ дорівнює 0.00001. Мінімізація функції L виконувалась за допомогою оптимізатора Adam з кроком навчання 0.001.
На рисунку 2.21 зображено криву зміни функції втрат для навчального і тестового набору.

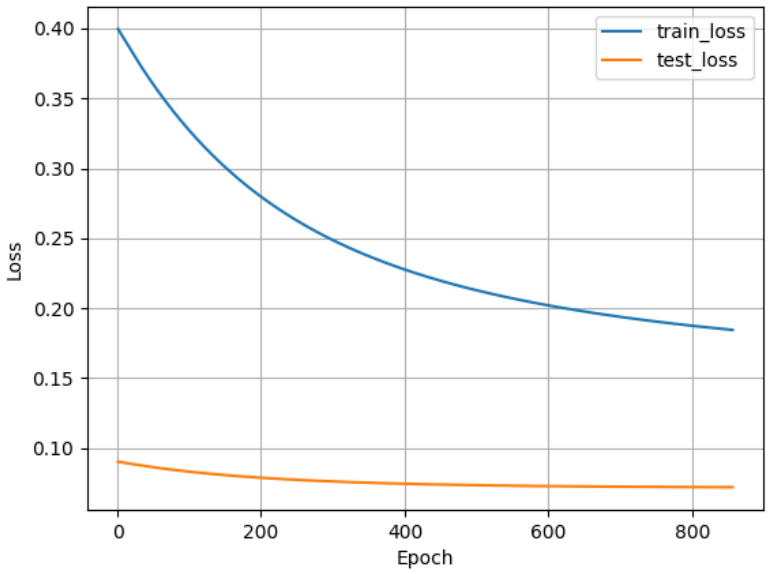


Рисунок 2.21 Крива зміни функції втрат.

Можна помітити, що крива втрат тестової загалом менше, ніж для навчальної. У кінці навчання моделі значення функції втрат для навчального

набору дорівнювало 0.18, а для тестового 0.072. Це явище можна обґрунтувати тим, що кількість ненульових значень у тестовому наборі набагато менше, ніж у навчальному – 334 тисячі проти 50 тисяч. Різниця у 6 разів дозволяє зменшити середнє квадратичне відхилення за рахунок того, що там може міститись менше викидів, таких як, наприклад, справжній залишений рейтинг дорівнює 5, а модель розрахувала 1. Корінь середньої квадратичної похибки для навчального набору дорівнює 0.51, а для тестового 0.21.

Отже, перейдемо тепер до розрахунку метрики precision@k, де k дорівнює 10. Значення цієї метрики буде усередненим по користувачам, які мають багато оцінок. Це зроблено з тою метою, що коли у користувача мало оцінок, наприклад 15, то випадковим перемішуванням мінімальним результатом precision@10 буде 0.5, тому що максимум, який можна втратити з топ-10 елементів - це 5 елементів, які знаходяться від 11 до 15 позиції. Таким чином, чим більше значення k, тим більше буде влучність, тому що зменшується строгість.

На рисунку 2.22 зображено правдивий список топ-10 елементів з ідентифікатором фільму і оцінкою схожості. На рисунку 2.23 зображено відсортований список за прогнозованою оцінкою, який вже об’єднали з правдивою оцінкою схожості. Ми можемо відслідкувати, який з елементів було рекомендовано в топ 10 за допомогою ідентифікаторів фільму.

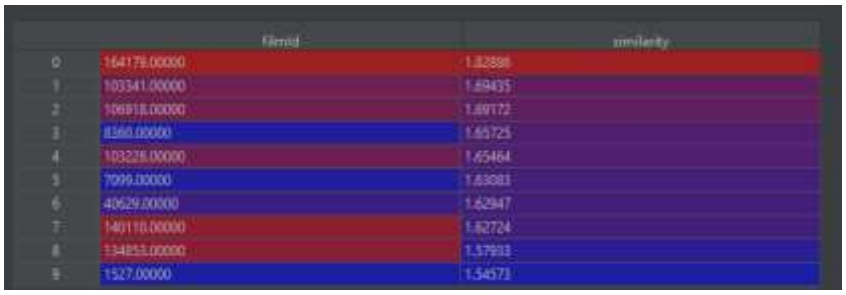
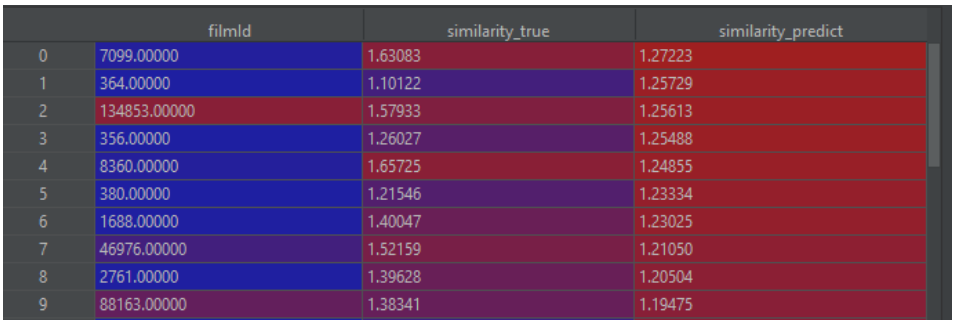


Рисунок 2.22 Правдивий (ground truth) список топ-10 елементів.



	filmId	similarity_true	similarity_predict
0	7099.00000	1.63083	1.27223
1	364.00000	1.10122	1.25729
2	134853.00000	1.57933	1.25613
3	356.00000	1.26027	1.25488
4	8360.00000	1.65725	1.24855
5	380.00000	1.21546	1.23334
6	1688.00000	1.40047	1.23025
7	46976.00000	1.52159	1.21050
8	2761.00000	1.39628	1.20504
9	88163.00000	1.38341	1.19475

Рисунок 2.23 Відсортований список оцінок за прогнозованою оцінкою.

Порівнявши два рисунки, можна відмітити, що фільми з ідентифікаторами 7099, 134853, 8360 були спрогнозовані рекомендаційою системою в топ 10.

Отже, розглянувши методологію розрахунку, перейдемо до усередної оцінки за наборами даних: для навчальної вибірки значення $precision@10$ буде дорівнювати 0.17, для тестової – 0.16. Це значить, що на 10 елементів, які рекомендує система, існує мінімум 1, який заінтересує користувача.

Перейдемо до наступної моделі, а саме SVD ++.

2.5.4 SVD ++

Ознайомившись з теорію про модель SVD++ у розділі 1.4.3, приступимо до форматування нашої архітектури у класі моделі, побудованної на базі Tensorflow. Для створення цієї моделі треба розрахувати середній ненульовий рейтинг у наборі даних. Також модель потребує окремі змінні для зміщення, яке окремо вносить фільм і користувач. Рейтингова оцінка буде розраховуватись за наступною формулою:

$$\hat{r}_{ui} = \mu + b_i + b_u + p_u q_i^T, \quad (2.6)$$

де μ – середнє значення балів, виставлених по всіх пунктах всіма користувачами; b_i – зміщення, яке вносить елемент i; b_u – зміщення, яке вносить користувач u, p_u – вектор ознак користувача, q_i – вектор ознак елементу.

Як можна побачити, модель має зміщення для фільму або користувача, що дозволить вирішити проблему холодного старту, коли ознаки фільму або користувача невідомі.

Функція втрат виглядає наступним чином:

$$L = \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \|p_u\|^2 + \|q_i\|^2) \qquad (2.7)$$

де μ – середнє значення балів, виставлених по всіх пунктах всіма користувачами; b_i – зміщення, яке вносить елемент i; b_u – зміщення, яке вносить користувач u, p_u – вектор ознак користувача, q_i – вектор ознак елементу, λ – коефіцієнт регуляризації.

Коефіцієнт регулярізації λ дорівнює 0.005. Мінімізація функції L виконувалась за допомогою оптимізатора Adam з кроком навчання 0.001.
На рисунку 2.24 зображено криву зміни функції втрат для навчального і тестового набору.

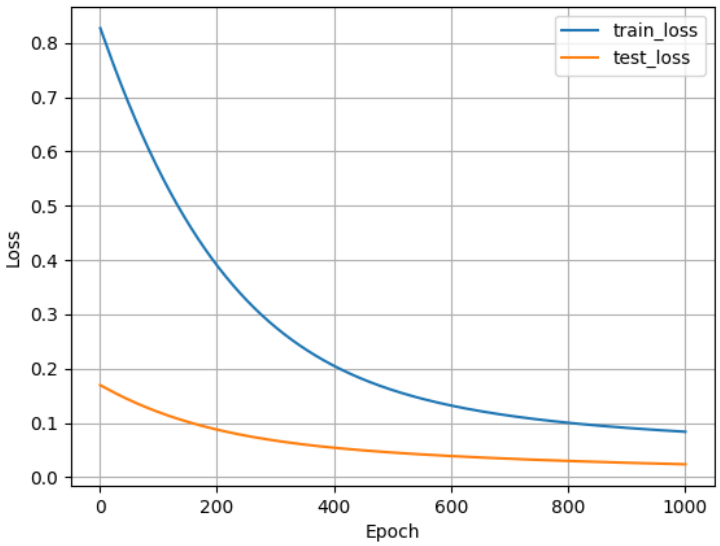


Рисунок 2.24 Крива зміни функції втрат.

У кінці навчання моделі значення функції втрат для навчального набору дорівнювало 0.08, а для тестового 0.03. Природа поведінки функції втрат для тестової виборки та ж сама, що і для Funk SVD – корінь середньої квадратичної похибки для навчальної вибірки дорівнює 0.36 , а для тестової

0.15. За рахунок меншої кількості можливих викидів значення функції втрат, яка базується на середній квадратичній похибці, є меншим.

Отже, перейдемо тепер до розрахунку метрики precision@k, де k дорівнює 10. Значення цієї метрики буде усередненим по користувачам, які мають більше 70 відгуків, незалежно від типу набору даних (навчальний, тестовий). Це зроблено з тою метою не завищувати значення метрики викидами з малою кількістю відгуків.

Усереднена оцінка $precision@10$ для навчального набору даних буде дорівнювати 0.4, для тестової – 0.4. Це значить, що на 10 елементів, які рекомендує система, існує мінімум 4, які зацікавлять користувача.

Перейдемо до наступної моделі, а саме модернізованої Funk SVD.

2.5.5 Funk SVD з тегами

Третім варіантом моделі для рекомендаційної системи буде модель Funk SVD з додатковою інформацією, а саме з використанням Explicit Feedback у вигляді тегів. Мотивацією до створення цієї моделі було обмеження знань про користувача, в той час, як про фільм було відомо жанри, рік випуску.

Як можна пригадати з розділу 2.4, набір вхідних даних містить таблицю з тегами, які залишали користувачі разом з рейтинговою оцінкою. Також розробник набору даних надав таблицю, яка містить оцінку в інтервалі (0;1), що показує наскільки коректно тег описує фільм.

Ідея моделі, яка представлена у цьому розділі, полягає у тому, щоб за допомогою тегів оцінити схожість між користувачем і фільмом. Оскільки користувач самостійно обирає теги, якими описати фільм, це допоможе врахувати його додаткові ознаки, які неможливо виразити за допомогою набору з 25 ознак.

Для того, щоб виразити усі теги для кожного користувача і фільму у вигляді матриці взаємодії буде потрібна не одна матриця (щоб врахувати який саме тег було використано). Цей спосіб потребує великих ресурсів

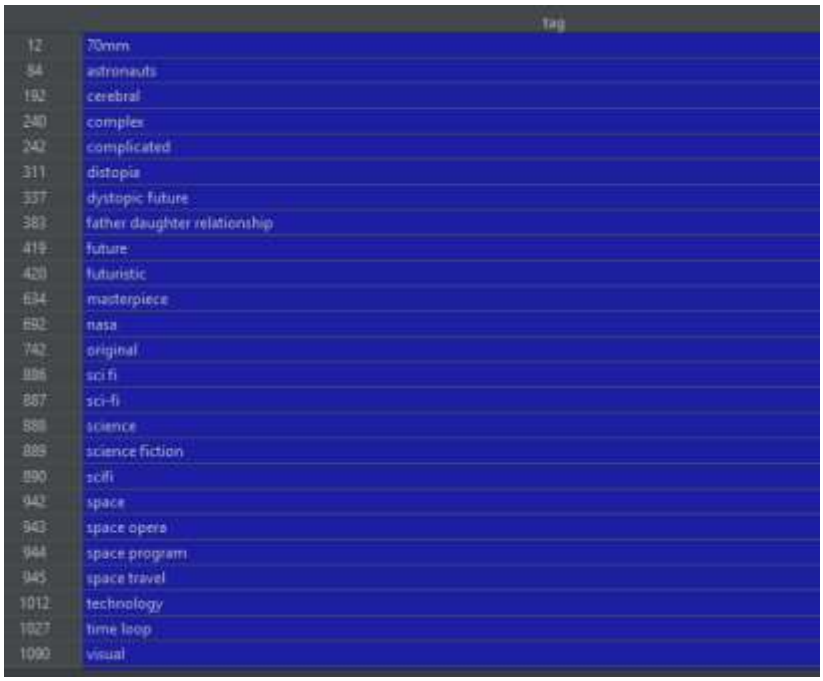
пам’яті, тому було вирішено залучити методику обробки слів з галузі обробки природньої мови у машинному навчанні (Natural Language Proccesing). Методика полягає у тому, щоб кожне слово перевести з текстового вигляду у вектор ембедингів, наприклад, ембединги word2vec або GloVe (див. розділ 1.3.3). Далі, за допомогою властивості додавання ембедингів, яка дозволяє об’єднувати семантичний зв’язок декількох словосполучень, ми можемо знаходити слова-синоніми до цих словосполучень. Для того, щоб знайти синоніми треба застосувати функцію косинусної схожості (див. розділ 1.2.1) до кожного слова у словнику і обрати з відсортованого за спаданням списку перші n слів.

Таким чином, ми можемо вибрати усі теги, які найбільше відповідають фільму (перевищують поріг відповідності, наприклад, для цього завдання було обрано поріг 0.9), розбити теги на токени, знайти для них ембедінги і виконати операцію додавання.

Розрахуємо слова синоніми для знайденого вектору ембедінгу і порівняємо їх за семантичним сенсом з тегами. На рисунку 2.25 зображено список тегів для фільму Interstellar, які мають коефіцієнт релевантності більше 0.9.

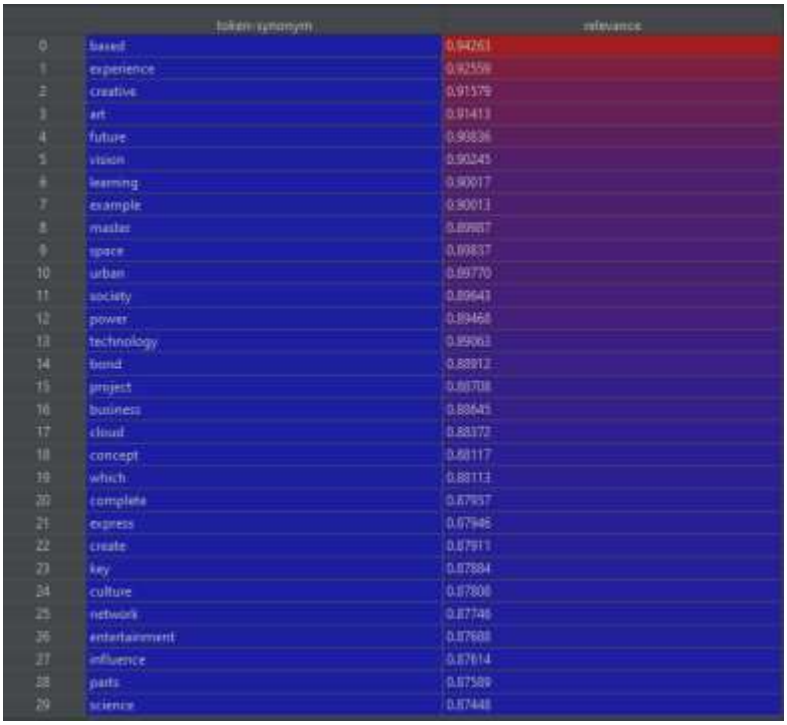
Для знаходження ембедингів тегів було використано модель GloVe з бібліотеки gensim.[12] Ця модель була навчена на великому наборі текстів з Twitter і має різновид, а саме різний розмір векторів кодування: 25,50 і 100. Щоб з’ясувати, який саме розмір вектору кодування обрати, виведемо слова синоніми з оцінкою схожості. Проте, як було підмічено, не всі користувачі залишали теги, тому для таких користувачів вектор тегів буде нульовим і відповідно не буде мати синонімів. Програмний код вибору і створення ембедингів тегів можна побачити у додатку Е і Є відповідно.

На рисунку 2.26 зображено слова синоніми для ембедингів тегів з розміром вектору 25 значень.



	tag
12	70mm
84	astronauts
192	cerebral
240	complex
242	complicated
311	distopia
337	dystopic future
383	father daughter relationship
419	future
420	futuristic
634	masterpiece
682	nasa
742	original
886	sci fi
887	sci-fi
888	science
889	science fiction
890	scifi
942	space
943	space opera
944	space program
945	space travel
1012	technology
1027	time loop
1090	visual

Рисунок 2.25. Теги до фільму Interstellar.



	token synonym	relevance
0	based	0.94261
1	experience	0.92559
2	creative	0.91579
3	art	0.91413
4	future	0.90836
5	vision	0.90245
6	learning	0.90017
7	example	0.90013
8	master	0.89887
9	space	0.89837
10	urban	0.89770
11	society	0.89643
12	power	0.89468
13	technology	0.89063
14	bond	0.88912
15	project	0.88708
16	business	0.88645
17	cloud	0.88372
18	concept	0.88117
19	which	0.88113
20	complete	0.87957
21	express	0.87946
22	create	0.87911
23	key	0.87884
24	culture	0.87808
25	network	0.87746
26	entertainment	0.87688
27	influence	0.87614
28	parts	0.87589
29	science	0.87448

Рисунок 2.26. Токени синоніми до комбінованого ембедингу (25) тегів фільму Interstellar.

На рисунку 2.27 зображено слова синоніми для ембедингів тегів з розміром вектору 50 значень.

	token-synonym	relevance
0	space	0.86444
1	based	0.85829
2	experience	0.85539
3	creative	0.84238
4	which	0.83402
5	project	0.83179
6	create	0.82784
7	book	0.82372
8	technology	0.82110
9	future	0.82056
10	the	0.81595
11	common	0.81559
12	action	0.81516
13	tech	0.81514
14	new	0.81513
15	any	0.81378
16	interesting	0.81343
17	art	0.81156
18	find	0.81115
19	life	0.80922
20	reality	0.80902
21	of	0.80839
22	science	0.80753
23	power	0.80665
24	our	0.80597
25	business	0.80551

Рисунок 2.27. Токени синоніми до комбінованого ембедингу (50) тегів фільму Interstellar.

Розглядаючи слова на рисунку 2.26 і 2.27 можна побачити, що серед синонімів присутні слова «space», «technology», «future», «science», які також знаходяться у тегах до фільму. Отже, можна вважати, що об'єднання тегів за допомогою кодування ембедингами дає жаданий результат.

Отже, опрацювавши теги користувачів і фільмів, ми отримаємо матриці ембедингів розміром 1214 на 50 і 1557 на 50 відповідно. Запишемо формулу отриманої моделі:

$$\hat{r}_{ui} = p_u * (q_i + b_i) + \widehat{H}_u \widehat{W}_i, \tag{2.8}$$

де $p_u -$ вектор ознак користувача u, $q_i -$ вектор ознак фільму i, $b_i -$ зміщення ознак фільму i, $\widehat{H}_u -$ нормалізований за L2 вектор ембедингів тегів користувача u, $\widehat{W}_i -$ нормалізований за L2 вектор ембедингів тегів фільму i.

L2 нормалізація виконується за наступною формулою:

$$\hat{v} = \frac{v}{\|v\|^2} = \frac{v}{\sqrt{\sum v_i^2}} \tag{2.9}$$

де $\hat{v} -$ це нормалізований вектор, $v -$ вектор ознак

$$cosine_{similarity(H_u W_i)} = \frac{H_u W_i}{\|H_u\|^2 \|W_i\|^2} \tag{2.10}$$

$H_u -$ вектор ембедингів тегів користувача u, $W_i -$ вектор ембедингів тегів фільму i.

Значення виразу у формулі 2.10 лежить на відрізку [-1;1]. Для перевірки того, що існує така пара елементів, яка буде від'ємною, розглянемо значення розрахунків на рисунку 2.28. Як можна побачити, ситуація з від'ємним результатом скалярного множення існують, що значить , що вектори ембедингів направлені в різні сторони.

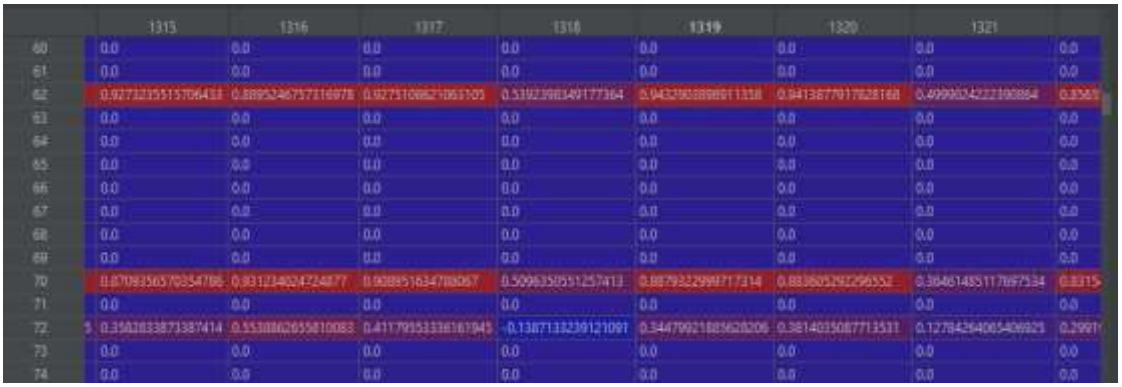


Рисунок 2.28. Розрахунок для формули 2.9.

Для навчання моделі використовувалась наступна функція втрат:

$$L = \frac{1}{2} \sum_{(u,i) \in S} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2} * (\|p_u\|^2 + \|b_i\|^2) \tag{2.11}$$

де $b_i -$ зміщення, яке вносить елемент $i, p_u -$ вектор ознак користувача, $\lambda -$ коефіцієнт регуляризації, $r_{ui} -$справжнє значення рейтингу, $\hat{r}_{ui} -$ прогнозоване значення рейтингу.

Коефіцієнт регулярізації λ дорівнює 0. 00005. Мінімізація функції L виконувалась за допомогою оптимізатора Adam з кроком навчання 0.001.
На рисунку 2.29 зображено криву зміни функції втрат для навчального і тестового набору.

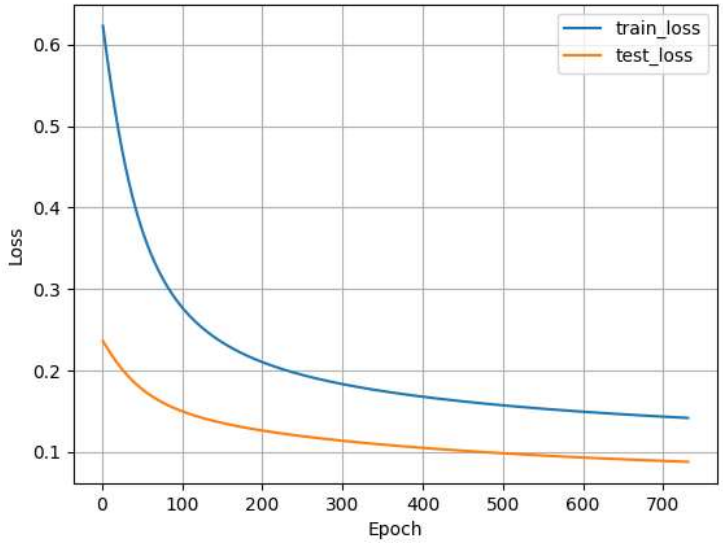


Рисунок 2.29. Функція втрат для модернізованої моделі Funk SVD.

У кінці навчання моделі значення функції втрат для навчального набору дорівнювало 0.14, а для тестового 0.08. Причина, за якої значення втрат для тестового набору нижче, та сама, що і для Funk SVD і SVD++. Модель залежить від значень-викидів у прогнозах, тому корінь середнього квадратичного відхилення дорівнює 0.35 для навчального набору і 0.14 для тестового. Навчальний набор в 6 раз більше тестового і в теорії містить більше значень-викидів, що призводить до збільшення метрики RMSE.

Отже, тепер перейдемо до розрахунку метрики precision@k, де k дорівнює 10. Усереднена оцінка precision@10 для навчального набору даних буде дорівнювати 0.4, для тестової – 0.38. Це значить, що на 10 елементів, які рекомендує система, існує мінімум 5, які зацікавлять користувача.

2.5.6 Порівняння моделей

Для порівняння трьох моделей, які було представлено в цьому розділі, використаємо метрику precision@k при різних значеннях k, а саме 10, 20, 50. Програмний код для тестування моделей можна знайти у додатку Ґ. Чим більше діапазон k, тим вища вірогідність, що в рекомандації попаде фільм, на який звернув увагу користувач. Значення метрик зведено в таблиці (див. таблицю 2.1, 2.2), які діляться на навчальну і тестову вибірку.

Таблиця 2.1

Значення метрики precision@k при k=10,20,50 для навчального набору даних

k	Funk_SVD	SVD++	SVD_with_tags
10	0,17	0,4	0,4
20	0,22	0,55	0,58
50	0,42	0,708	0,69

Таблиця 2.2

Значення метрики precision@k при k=10,20,50 для тестового набору даних

k	Funk_SVD	SVD++	SVD_with_tags
10	0,16	0,4	0,38
20	0,24	0,57	0,53
50	0,42	0,709	0,66

Аналізуючи таблиці, можна помітити, що модель Funk SVD значно програє обом моделям, а модель з тегами трохи гірше за модель зі зміщеннями. Отже, кращою моделлю виявилась SVD ++.

В додаток до таблиць з метриками, подивимось на справжні рейтингові оцінки (truth_rating) і ті, що прогнозують моделі. Список був відсортований за спаданням рейтингової оцінки SVD++ (див. рисунок 2.30-2.31).

title	truth_rating	predict_rating_Funk_SVD	predict_rating_SVD_plus	predict_rating_SVD_with_tags
3:10 to Yuma	4.00000	3.93918	5.00000	3.13979
48 Hrs.	4.00000	4.61129	5.00000	2.52998
Beverly Hills Cop	4.00000	4.61129	5.00000	2.93730
Big	4.00000	3.95732	5.00000	2.92689
Big Chill, The	2.00000	3.02416	5.00000	3.12174
Blazing Saddles	4.00000	2.58652	5.00000	3.19927
Blood Simple	4.00000	2.68458	5.00000	3.38811
Butch Cassidy and the Sundance Kid	4.00000	2.00119	5.00000	3.61698
Dark Knight, The	5.00000	4.01000	5.00000	3.71774
Evil Dead II (Dead by Dawn)	4.00000	3.96942	5.00000	3.06430
Fargo	4.00000	4.47138	5.00000	3.69005
Fight Club	4.00000	3.88605	5.00000	3.73258
Fish Called Wanda, A	4.00000	3.11304	5.00000	3.26941
Godfather, The	4.00000	2.51815	5.00000	3.84228
Godfather: Part II, The	4.00000	2.51815	5.00000	3.70151
Grosse Pointe Blank	4.00000	3.49936	5.00000	3.07844
Hoosiers (a.k.a. Best Shot)	3.00000	2.25558	5.00000	3.20520
Lethal Weapon	4.00000	4.61129	5.00000	2.90169
Midnight Run	4.00000	4.66081	5.00000	3.41343
Miller's Crossing	4.00000	3.29709	5.00000	3.39759
My Neighbor Totoro (Tonari no Totoro)	4.00000	2.81689	5.00000	3.25522

Рисунок 2.30. Порівняння прогнозів між моделями.

title	truth_rating	predict_rating_Funk_SVD	predict_rating_SVD_plus	predict_rating_SVD_with_tags
Hunt for Red October, The	3.00000	2.92571	4.01246	3.30882
Casablanca	4.00000	1.65822	4.00668	3.62182
Ocean's Thirteen	4.00000	2.92511	4.00384	2.93112
Notebook, The	4.00000	2.61004	4.00354	3.09347
Birdman: Or (The Unexpected Virtue of Igno...	4.00000	3.36063	4.00345	3.65166
Labyrinth	3.00000	2.07627	3.99598	2.66980
Sherlock Holmes	4.00000	4.10712	3.99185	2.84874
Life Aquatic with Steve Zissou, The	4.00000	3.61698	3.99094	2.94268
13 Going on 30	4.00000	3.56595	3.98855	2.45102
Michael Clayton	3.00000	2.83623	3.98379	3.28957
1408	4.00000	3.41092	3.98037	3.19751
District 9	4.00000	3.13432	3.97872	3.34031
Remember the Titans	4.00000	2.04087	3.97840	3.31719
Scream	4.00000	3.89511	3.97486	3.16578
Death Proof	4.00000	4.87242	3.96981	3.01890
Chicken Run	4.00000	3.39927	3.96928	2.94412
Elf	4.00000	3.34337	3.96473	2.74591
Her	4.00000	3.20639	3.96099	3.52595
Wedding Crashers	4.00000	3.20195	3.95947	2.79121
Speed	3.00000	2.87467	3.95764	2.95685
Dirty Dancing	3.00000	2.40707	3.95471	2.77667

Рисунок 2.31. Порівняння прогнозів між моделями.

Орієнтиром для правильності прогнозованої рейтингової оцінки буде стовпчик «truth_rating» . Можна побачити, що модель Funk SVD часто сильно

недооцінює фільми, схожа поведінка і у моделі с тегами. Також можна помітити таку ознаку, що в списку знаходяться фільми-сіквели, тобто фільми, де один є продовженням іншого.

2.6 Ознаки користувача

Отже, повернімося до пункту 2.5.1 цього розділу, де згадувалось поняття latent features. Перед моделями матричної факторизації поставлено завдання підібрати значення матриці ознак користувача, при цьому використовувати фіксовану матрицю ознак фільмів. Такий підхід дозволить нам, використовуючи ознаки з відомими назвами, визначити відношення користувача до окремої ознаки.

Яскравим прикладом, де ознаки користувача будуть виражені в повну міру - це буде перша модель Funk SVD. Це відбувається за рахунок того, що градієнтний спуск для функції втрат не розпиляється на декілька змінних, а направлений лише на матрицю ознак користувача. Наприклад, в другій моделі неможливо повністю оцінити цей фактор, тому що в формулі присутній середній бал, зміщення користувача і фільму.

Розглянемо вектори ознак користувача і усереднені значення ознак для фільмів, які були високо оцінені цим користувачем, розрахуємо кореляцію Пірсона і спробуємо дати тлумачення значенням (див. таблицю 2.3).

Порівнюючи значення векторів між моделями, можна побачити, що вектор, виведений з моделі SVD++, зовсім не корелюється з вектором ознак фільмів, а також відрізняється від інших моделей. Внизу таблиці знаходяться розраховані коефіцієнти кореляції Пірсона для ознак фільмів і векторами окремої моделі. Можна побачити, що саме SVD++ має найменшу кореляцію.

Таким чином, ми побачили, що нам вдалося побудувати моделі, яка зможе розраховувати ознаки користувача, в якій ми знаємо значення кожної з

них, а саме дізнатися якому жанру і року випуску користувач надає більшу перевагу.

Таблиця 2.3

Порівняння ознак користувача між моделями і ознаками фільмів з високою оцінкою

features	film_embed	user_embed_Funk_SVD	user_embed_SVD++	user_embed_SVD_with_tags
Action	0,61044	0,64321	0,01562	0,42946
Adventure	0,5802	0,86694	-0,06192	0,37327
Animation	0,41458	0,25527	-0,01042	0,28517
Children	0,42868	0,21683	-0,02869	0,28445
Comedy	0,60553	0,59413	0,04512	0,40247
Crime	0,4714	0,14808	0,06532	0,36886
Documentary	0,30077	0,23219	0,68315	0,23406
Drama	0,84271	0,6783	-0,18851	0,50346
Fantasy	0,40611	0,40349	0,39985	0,31254
Film-Noir	0,28869	0,08263	-0,42996	0,34421
Horror	0,32333	0,03064	0,14212	0,23519
IMAX	0,37443	0,2119	-0,51561	0,32494
Musical	0,32599	0,05128	-0,58589	0,15597
Mystery	0,40679	0,21315	-0,32502	0,26371
Romance	0,47929	0,33546	0,17157	0,25018
Sci-Fi	0,44767	0,53123	0,04775	0,23633
Thriller	0,54539	0,39854	0,08613	0,24457
War	0,34391	0,18273	0,0162	0,21295
Western	0,30181	0,02041	-0,71857	0,21308
age_1950s-70s	0,37147	0,17489	0,00609	0,27593
age_1980s	0,33536	0,1171	-0,25129	0,15951
age_1990s	0,49057	0,25415	0,11268	0,3118
age_2000s	0,63012	0,66482	0,3293	0,30201
age_2010s	0,58906	0,8881	0,06389	0,39
age_older 1950s	0,32283	0,19549	0,61036	0,17964
Коеф. кореляції Пірсона		0,8274	0,1172	0,7751

Приведемо наглядний приклад, як ці ознаки користувача допомагають визначити важливість окремої ознаки. Наприклад, візьмемо вектор не з 25 ознак, а з 5, серед яких є: Action, Comedy, Thriller, age_2010s,age_1980s. Візьмемо два фільми і запишемо його ознаки у таблицю 2.4, а ознаки користувача у таблицю 2.5.

Таблиця 2.4

Ознаки фільмів

	Action	Comedy	Thriller	age_2010s	age_1980s
film_1	1	1	0	1	0
film_2	1	1	0	0	1

Ці фільми мають ідентичні жанри, але відрізняються за роком випуску.

Таблиця 2.5

Ознаки користувача

	Action	Comedy	Thriller	age_2010s	age_1980s
user_1	0,4	0,2	0,6	0,2	0,5

Помножимо скалярно вектори ознак і порівняємо значення.

$$user_1, film_1 = 0.4 * 1 + 0.2 * 1 + 0.6 * 0 + 0.2 * 1 + 0.5 * 0 = 0.8$$

$$user_1, film_2 = 0.4 * 1 + 0.2 * 1 + 0.6 * 0 + 0.2 * 0 + 0.5 * 1 = 1.1$$

Отже, як видно, другий фільм, який був випущений раніше, ніж перший, має більше значення скалярного добутку. Таким чином, цей вектор користувача дозволить нам отримати його характеристику, а саме знати які фільми йому подобаються. Ці знання також можуть допомогти при оптимізації процесів онлайн-кінотеатру, підвищити обізнаність власною аудиторією, стати інформацією для інших досліджень тощо. Також такі розробки, в результаті яких знаходиться нова інформація, можуть зекономити ресурси підприємства і посприяти виявленню проблеми або загрози, на які не звертали увагу раніше.

2.7 Висновок до розділу

Отже, ми визначилися з рекомендаційною системою, яка найкраще відображає інтереси користувачів, а саме модель SVD++. Вона була обрана через те, що показувала найвищий результат за метрикою precision@k, яка дозволяє оцінити частку рекомендованих елементів, які визвали інтерес у користувача. Модель правильно рекомендувала 4 елементи з 10 запропонованих. Зі збільшенням розміру списку видачі рекомендацій відповідно зростала кількість елементів, які зацікавили користувача, а саме з 20 виданих елементів зацікавили вже 11, а з 50 – вже 35.

Операція матричної факторизації дозволила перенести ознаки фільмів на користувачів, а саме дізнатись ступінь інтересу користувача до наявних жанрів і десятиріччя, у якому було випущено фільм. Але, як було вказано у розділі 2.6, модель SVD++ погано виконала цю процедуру, тому для того, щоб виконати це завдання, при цьому не сильно жертвувати якістю рекомендацій, можна використати модель Funk SVD з тегами. Використання тегів дозволило знайти додаткову інформацію про користувачів, яка є дуже цінною в цій ситуації, тому що інших джерел інформації про користувача не існувало.

Розробка рекомендаційної системи дозволила вирішити питання не лише покращення сервісу онлайн кінотеатру, а також знайти інформацію про користувачів, маючи при цьому лише історію їхньої взаємодії з фільмами та коментарі. Ця інформація може допомогти при оптимізації процесів онлайн-кінотеатру, підвищити обізнаність власною аудиторією, стати інформацією для інших досліджень тощо. Також такі розробки, в результаті яких знаходиться нова інформація, дозволяє зекономити ресурси підприємства і допоможе виявити проблеми або загрози, на які не звертали увагу раніше.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто проблему створення рекомендаційної системи підбору фільмів для онлайн-кінотеатру. Використання автоматизованої системи, яка враховує особливості користувача, дозволяє приваблювати і утримувати аудиторію, на відміну від сервісів, які видають контент у випадковому порядку.

У першому розділі було розглянуто загальні відомості цієї галузі, описано базові стратегії, використання яких дозволяє створити мінімально обізнану рекомендаційну систему, використовуючи стратегію memory-based. Проблеми цієї стратегії полягають в неможливості додати окремі ознаки, обмеженості даних, які використовуються для розрахунку.

Для вирішення цієї проблеми запропоновано створити model-based рекомендаційну систему, особливостями використання якої є можливість перенесення ознак з групи товару на групу користувачів. Цей підхід дозволив додати додаткову інформацію у вигляді зміщень окремих елементів, які можуть вирішити проблему холодного старту, а саме модель SVD++.

Окрім зміщень, у цій роботі було розглянуто модель з використанням текстових повідомлень користувача, які допоможуть враховувати їхні особливості. Для обробки текстових повідомлень було застосовано метод GloVe з галузі обробки природньої мови за допомогою машинного навчання.

Для побудови моделей було використано бібліотеку Tensorflow, яка дозволила спрямувати усі зусилля на розробку архітектури і логіки моделі, а методи і функції, які обслуговують її, залишити вбудованим функціям.

У другому розділі описано процес обробки об’єкту дослідження у вигляд, який може бути опрацьований моделлю машинного навчання. Використовуючи підготовленні дані, було розроблено три архітектури моделей, серед яких: Funk SVD, SVD++, Funk SVD з тегами. За допомогою метрики precision@k зі значеннями k=10, 20, 50 було порівняно якість рекомендації кожної з моделей.

За результатами порівняння було виявлено, що модель SVD++ показує найкращу продуктивність, але при цьому погано виконує задачу перенесення ознак фільмів на ознаки користувача, оскільки отримане значення кореляції Пірсона для векторів фільмів і користувача дорівнювало лише 0.11. Виконання цієї задачі не було першочерговим, але дозволило б зменшити витрати на додаткові дослідження і сформувати знання про користувача, які можуть бути стартовою точкою для інших досліджень.

Інша модель – Funk SVD з тегами, яка використовувала теги користувача для підвищення врахування особливостей, змогла виконати цю задачу переносу. Їй це вдалося зробити через відсутність додаткових зміщень у формулі розрахунку. Перевагою стало також те, що значення метрик precision@k були незначно менше від моделі SVD++.

Розробка рекомендаційної системи дозволила вирішити питання не лише покращення сервісу онлайн кінотеатру, а також знайти інформацію про користувачів, яка не лежала на поверхні, використовуючи для цього лише історію їхньої взаємодії з фільмами та коментарі. Отримана інформація може допомогти при оптимізації процесів онлайн-кінотеатру, підвищити обізнаність власною аудиторією, стати інформацією для інших досліджень тощо. Також такі розробки, в результаті яких знаходиться нова інформація, дозволяє зекономити ресурси підприємства і допоможе виявити проблеми або загрози, на які не звертали увагу раніше.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to collaborative filtering, URL : https://www.analyticsvidhya.com/blog/2022/02/introduction-to-collaborative-filtering/#h2_6 (дата звернення: 30.11.2024)

2. Similarity metrics for vector search, URL: https://zilliz.com/blog/similarity-metrics-for-vector-search (дата звернення: 30.11.2024)

3. Collaborative Filtering, URL: https://medium.com/@toprak.mhmt/collaborative-filtering-3ceb89080ade (дата звернення: 30.11.2024)

4. Different type of recommender systems, URL: https://medium.com/@itua/different-types-of-recommender-systems-a5545c8a074a (дата звернення: 30.11.2024)

5. Introduction to recommender systems 2. Deep neural network based recommendation systems, URL: https://towardsdatascience.com/introduction-to-recommender-systems-2-deep-neural-network-based-recommendation-systems-4e4484e64746 (дата звернення: 30.11.2024)

6. Trabelsi, F., Khtira, A., El Asri B. Hybrid Recommendation Systems: A State of Art [Електронний ресурс]/ Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021) – 2021 – с.281-288 – Режим доступу: https://pdfs.semanticscholar.org/455f/73531e3bda0e2551687f3b923a4c78dc24ff.pdf

7. One-hot-encoding, URL: https://www.deepchecks.com/glossary/one-hot-encoding (дата звернення: 30.11.2024)

8. T. Mikolov, K.Chen, G.Corrado, J.Dean. Efficient Estimation of Word Representations in Vector Space. 2013. с. 12. DOI: https://doi.org/10.1109/ICOSEC49089.2020.9215319

9. J.Pennington, R.Socher, C.D.Manning GloVe: Global Vectors for Word Representation. 2014. DOI: https://doi.org/10.3115/v1/D14-1162

10. Matrix factorization in recommender systems, URL: https://towardsdatascience.com/matrix-factorization-in-recommender-systems-3d3a18009881 (дата звернення: 30.11.2024)

11. Tensorflow (recommender systems), URL: https://www.tensorflow.org/api_docs/python/tf (дата звернення: 30.11.2024)

12. Gensim, GloVe embeddings URL: https://radimrehurek.com/gensim/ (дата звернення: 30.11.2024)

13. MovieLens Latest Datasets, URL: https://grouplens.org/datasets/movielens/ (дата звернення: 30.11.2024)

14. Метод K-Nearest Neighbours. Розбір без використання бібілотек та з використанням бібліотек, URL: https://habr.com/ru/articles/680004 (дата звернення: 30.11.2024)

15. Чудовий світ Word Embeddings: які вони бувають і для чого потрібні? URL: https://habr.com/ru/companies/ods/articles/329410 (дата звернення: 30.11.2024)

16. Кваліфікаційна робота магістра [Електронний ресурс] : методичні рекомендації для здобувачів ступеня магістра освітньо-професійної програми «Системний аналіз» зі спеціальності 124 Системний аналіз / уклад.: Т.А. Желдак, Т.В. Хом’як, А.В. Малієнко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 33 с.

17. Matrix factorization (recommender systems) URL: https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems) (дата звернення: 30.11.2024)

18. A Python implementation of LightFM, a hybrid recommendation algorithm, URL: https://github.com/lyst/lightfm (дата звернення: 30.11.2024)

19. NLP. Основи. Техніки Саморозвиток. Частина 1, URL: https://habr.com/ru/companies/contentai/articles/437008/ (дата звернення: 30.11.2024)

20. Pytorch (recommender systems), URL: https://pytorch.org/docs/stable/index.html (дата звернення: 30.11.2024)

21. Хом'як Т. В. Бази даних у професійних задачах аналітики [Електронний ресурс] : навч. наочн. посіб. / Т. В. Хом'як, К. С. Хабарлак, Д.М. Гаранжа; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». − Дніпро : НТУ «ДП», 2024. − 192с.

22. Самонавчання складних систем [Електронний ресурс] : методичні рекомендації до виконання практичних робіт для здобувачів ступеня магістра освітньо-професійної програми «Системний аналіз» зі спеціальності 124 Системний аналіз / Т.А. Желдак, К.С. Хабарлак, Д.М. Гаранжа ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 66 с. https://ir.nmu.org.ua/handle/123456789/167645

23. Молоканова, В. М., & Шевченко, Ю. О. (2024). Управління проектною командою. https://ir.nmu.org.ua/handle/123456789/167646

24. Кваліфікаційна робота магістра [Електронний ресурс] : методичні рекомендації для здобувачів ступеня магістра освітньо-професійної програми «Системний аналіз» зі спеціальності 124 Системний аналіз / уклад.: Т.А. Желдак, Т.В. Хом'як, А.В. Малієнко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 33 с. https://ir.nmu.org.ua/handle/123456789/167921

ДОДАТКИ

Додаток А

Відомості матеріалів кваліфікаційної роботи

		Позначення	Найменування	Кількість	Примітка
1					
2			Документація		
3					
4		САУ.КР.24.02.ПЗ	Пояснювальна записка	96	Формат А4
5					
6		САУ.КР.24.02.ДМ	Демонстраційні матеріали	20	Презентація на CD-R
7					
8		САУ.КР.24.02.КР	Копія роботи	1	Диск CD-R

					САУ.КР.24.02.ДА.ПЗ			
Зм	Лист	№ докум.	Підпис	Дата				
Розроб.		Говоруха			**Матеріали кваліфікаційної роботи**	Літ	Аркуш	Аркушів
К. розд.		Желдак						
Керівн.		Мінєєв				НТУ «ДП», 12; 124м-23-1		
Н. контр.		Хом’як						
Зав. каф.		Желдак						