

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Аналіз та імплементація алгоритмів для побудови
персоналізованої рекомендаційної системи»**

Виконала студентка групи ІПЗм-23-1
ЛЕЩИШИНА Вікторія Олегівна

Керівник роботи:
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«01» жовтня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЛЕЩИШИНА Вікторія Олегівна**

Керівник роботи: **ЛОБАНЧИКОВА Надія Миколаївна**

Тема роботи: **«Аналіз та імплементація алгоритмів для побудови персоналізованої рекомендаційної системи»**,

затверджена Наказом закладу вищої освіти від **«01» жовтня 2024 р., №520/с**

Вихідні дані для роботи: Об'єктом дослідження є персоналізовані рекомендаційні системи, які надають рекомендації на основі аналізу даних.

Предметом дослідження є використання алгоритмів, методів та підходів для розробки персоналізованої рекомендаційної системи.

Керівники / консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Керівник/ консультант	Завдання видав	Завдання прийняв
1	Лобанчикова Н. М.	10.10.2024	10.10.2024
2	Лобанчикова Н. М.	10.10.2024	10.10.2024
3	Лобанчикова Н. М.	10.10.2024	10.10.2024

РЕФЕРАТ

Індивідуальне завдання кваліфікаційної магістерської роботи полягає в дослідженні персоналізованих рекомендаційних систем, принципів їх роботи, методів та алгоритмів для їх реалізації, а також аналізування розроблених рекомендаційних систем.

Кваліфікаційна магістерська робота на тему «Аналіз та імплементація алгоритмів для побудови персоналізованої рекомендаційної системи» складається з теоретичного дослідження, реалізованих рекомендаційних систем чотирьох типів та пояснювальної записки. Пояснювальна записка до кваліфікаційної магістерської роботи складається зі вступу, трьох розділів, списку використаних джерел з 31 найменуванням, 43 рисунки, 2 таблиць та 6 додатків. Повний обсяг роботи становить 71 сторінка.

КЛЮЧОВІ СЛОВА: РЕКОМЕНДАЦІНА СИСТЕМА, АНАЛІЗ ДАНИХ, PYTHON, МАТРИЧНА ФАКТОРИЗАЦІЯ, ВЕКТОРИЗАЦІЯ ДАНИХ.

ABSTRACT

The individual task of the qualifying master's thesis is to study personalized recommender systems, the principles of their operation, methods and algorithms for their implementation, as well as to analyze the developed recommender systems.

The qualifying master's thesis on “Analysis and Implementation of Algorithms for Building a Personalized Recommender System” consists of a theoretical study, implemented recommender systems of four types, and an explanatory note. The explanatory note to the qualification master's thesis consists of an introduction, three chapters, a list of 25 references, 43 figures, 2 tables, and 6 appendices. The total volume of the work is 71 pages.

KEYWORDS: RECOMMENDER SYSTEM, DATA ANALYSIS, PYTHON, MATRIX FACTORIZATION, DATA VECTORIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1 ОГЛЯД РЕКОМЕНДАЦІЙНИХ СИСТЕМ ТА ПРИНЦИПІВ ЇХ РОБОТИ	9
1.1 Загальний огляд рекомендаційних систем	9
1.2 Колаборативна фільтрація.....	11
1.3 Контент фільтрація	15
1.3 Гібридні системи	20
Висновки до першого розділу.....	24
РОЗДІЛ 2 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНИХ АЛГОРИТМІВ.....	25
2.1 Підготовка та аналіз набору даних.....	25
2.2 Реалізація колаборативної фільтрації	30
2.3 Реалізація контент фільтрації	32
2.4 Реалізація гібридної системи	37
Висновки до другого розділу	38
РОЗДІЛ 3 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ СИСТЕМ.....	39
3.1 Визначення критеріїв оцінювання	39
3.2 Кількісний аналіз	40
3.3 Якісний аналіз	43
3.4 Подальша робота.....	45
Висновки до третього розділу	46
ВИСНОВКИ.....	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	50
ДОДАТКИ.....	53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CF – (англ. collaborative filtering) – колабораційна фільтрація.

CB – (англ. content-based filtering) – контент фільтрація.

SVD – (англ. singular value decomposition) – сингулярний розклад матриці.

TF-IDF – (англ. term frequency – inverse document frequency) – статистичний показник, який використовується для оцінки важливості слова у контексті.

RMSE – (англ. root mean square error) – корінь середньоквадратичної похибки.

MSE – (англ. mean square error) – середньоквадратична похибка.

ВСТУП

Актуальність теми. Останніми роками рекомендаційні системи стали невід’ємною частиною повсякденного життя. У сучасному цифровому світі інформаційний потік невинно зростає і користувачам стає все важче орієнтуватися в таких колосальних обсягах даних. Один Amazon має в своєму каталозі більше 600 мільйонів найменування, без впроваджені системи рекомендацій користувачам важко було б знайти необхідні продукти, а компанія неодмінно втратила б прибуток.

Рекомендаційні системи є однією з найпомітніших і найважливіших технологій сучасного цифрового середовища. Вони відіграють ключову роль у покращенні користувацького досвіду та залученні користувачів.

Завдяки стрімкому розвитку технологій штучного інтелекту та машинного навчання, точність цих системи лише зростає, що в свою чергу підвищує задоволення індивідуальні потреби користувачів. Однак такі системи потребують постійних модифікацій та покращень для того, щоб встигнути за користувачами. Вибір правильної системи рекомендацій може полегшити життя не лише звичайним людям, а й компаніям.

Мета і завдання дослідження. Метою дослідження є аналіз та імплементація алгоритмів для розробки персоналізованої рекомендаційної системи. Для досягнення встановленої мети необхідним є вирішення наступних взаємозалежних задач дослідження:

- проведення аналізу рекомендаційних систем та принципів їх роботи;
- проведення програмної реалізації рекомендаційних алгоритмів;
- проведення аналізу кількісних та якісних показників впровадження рекомендаційних алгоритмів.

Об’єктом дослідження є процеси і технології побудови та розробки персоналізованої рекомендаційної системи.

Предметом дослідження є алгоритми імплементації, методи та засоби для побудови персоналізованої системи рекомендацій.

Наукова новизна. У результаті проведеного дослідження було розроблено чотири типи рекомендаційних систем з використанням актуальних технологій машинного навчання. Проведений аналіз імплементованих алгоритмів визначив, що розроблений гібридний підхід показує найточніші результати.

Практичне значення полягає у систематизації інформації про рекомендаційні системи, їх реалізації та узагальнені даних, отриманих при аналізі.

РОЗДІЛ 1 ОГЛЯД РЕКОМЕНДАЦІЙНИХ СИСТЕМ ТА ПРИНЦИПІВ ЇХ РОБОТИ

1.1 Загальний огляд рекомендаційних систем

Рекомендаційні системи стали невід'ємною частиною сучасного цифрового досвіду, оскільки вони вирішують проблему інформаційного перевантаження в епоху, коли щодня генеруються величезні обсяги даних. У таких сферах, як електронна комерція, стрімінгові платформи та соціальні мережі, користувачам надається величезна кількість варіантів вибору, що ускладнює пошук релевантного контенту чи продуктів. Рекомендаційні системи використовують алгоритми для фільтрації та персоналізації цих варіантів, допомагаючи користувачам швидко знаходити товари або контент, які відповідають їхнім уподобанням. Це не тільки підвищує задоволеність користувачів, але й економить час, оскільки усуває необхідність просіювати нерелевантний або нецікавий матеріал.

З точки зору бізнесу, рекомендаційні системи є життєво важливими для покращення залучення клієнтів та збільшення доходів. Аналізуючи поведінку користувачів, їхні вподобання та попередні взаємодії, ці системи можуть запропонувати продукти чи послуги, які з більшою ймовірністю знайдуть відгук у кожного окремого клієнта. Такий персоналізований підхід часто призводить до підвищення конверсії та лояльності клієнтів, оскільки користувачі відчують, що їхні потреби та смаки розуміють. Крім того, рекомендаційні системи дозволяють компаніям ефективно використовувати свої дані, перетворюючи їх на дієву інформацію, яка може бути використана в маркетингових стратегіях, управлінні запасами та розробці продуктів.

Зростанню популярності рекомендаційних систем також сприяє прогрес у машинному навчанні та штучному інтелекті, який значно підвищив їхню точність і масштабованість [1]. Оскільки цифрові платформи конкурують між собою за безперебійний та персоналізований користувацький досвід, попит на надійні та ефективні системи рекомендацій продовжує зростати. Вони стали

наріжним каменем цифрової економіки, підвищуючи задоволеність користувачів і сприяючи успіху бізнесу в світі, де зростає конкуренція і де все більше керують дані.

Яскравим прикладом роботи рекомендаційних систем є платформа Amazon, яка вже давно використовує рекомендаційні алгоритми для збільшення продажів і підвищення рівня задоволеності клієнтів. Щоразу, коли користувач переглядає товар або завершує покупку, система оцінює його активність і порівнює з даними мільйонів інших користувачів. Використовуючи цю інформацію, Amazon генерує індивідуальні пропозиції, такі як: «Клієнти, які купили цей товар, також купили...» або «Рекомендуємо вам».

Ця стратегія не лише збільшує середні витрати одного користувача, але й сприяє створенню персоналізованого досвіду покупок, що заохочує до повторних звернень. Варто відмітити, що приблизно 35% доходу Amazon [2] надходить від рекомендаційних систем.

Загалом, рекомендаційні системи – це алгоритми, розроблені для надання персоналізованих пропозицій щодо товарів, які є релевантними для інтересів користувача. Ці системи генерують рекомендації, аналізуючи попередні дії та вподобання користувача. Як спеціалізований тип систем фільтрації інформації, вони використовують алгоритми для адаптації пропозицій на основі індивідуальних інтересів або активності.

Рекомендаційні системи можна поділити на фільтрацію вмісту (англ. content-based filtering), колаборативну фільтрацію (англ. collaborative filtering), яка в свою чергу поділяється на підхід на основі пам'яті (англ. memory-based filtering) і підхід на основі моделі (англ. model-based filtering) [3]. Типи рекомендаційних систем зображені на рис. 1.1.

Кожен з цих підходів має як свої переваги так і недоліки, і використовується для різних потреб. Для досягнення найкращого результату велика кількість компаній надає перевагу гібридним підходам, адже їх можна підналаштувати для власних потреб.

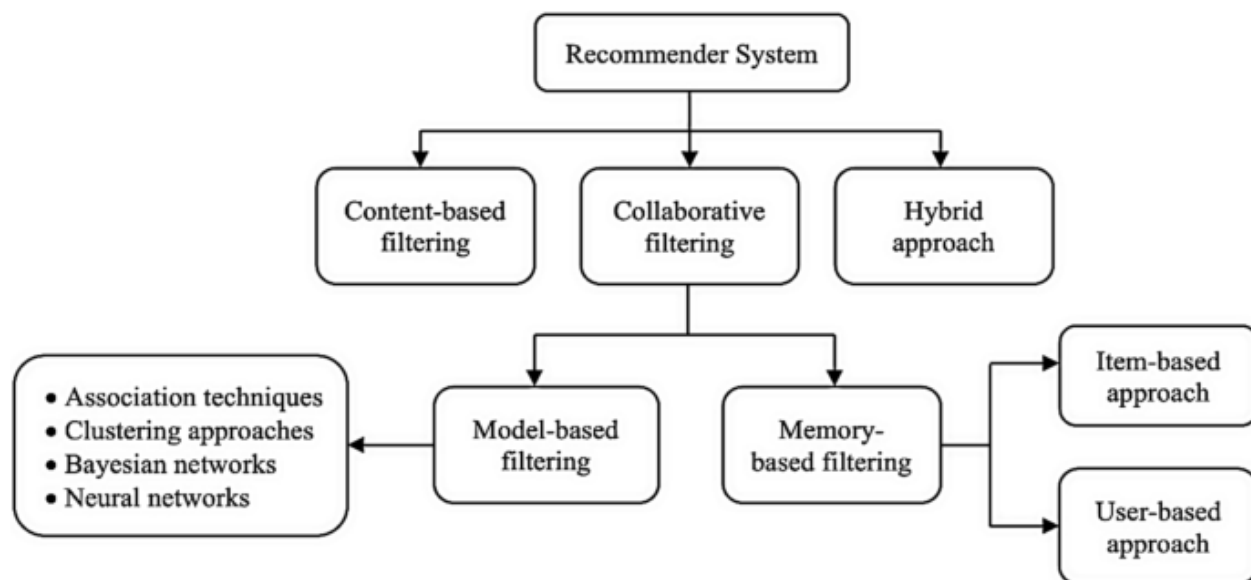


Рис. 1.1. Типи рекомендаційних систем

1.2 Колаборативна фільтрація

Принцип роботи колаборативної фільтрації (CF) полягає в аналізі поведінки багатьох користувачів. CF базується виключно на минулих взаємодіях між користувачами та об'єктами, записаних з метою створення нових рекомендацій [4]. Ці взаємодії зберігаються у так званій «матриці взаємодій користувачем-предмет».

Головна ідея полягає в тому, що минулі взаємодії між користувачем і об'єктом достатньо для виявлення схожих користувачів та об'єктів і створення прогнозів.

CF зазвичай поділяється на два підтипи: на основі пам'яті та на основі моделі [5]. У першому не передбачається ніякої латентної моделі. Алгоритми безпосередньо працюють з взаємодією користувача з об'єктом і для створення нових рекомендацій використовується алгоритм пошуку k-найближчих сусідів. Такий метод, теоретично, має низьку похибку, але високу дисперсію.

Підхід на основі пам'яті також поділяється на два підтипи: на основі користувача та на основі предмету. Для створення рекомендацій ці підходи фокусуються на схожості користувачів або предметів.

Перший тип рекомендує предмети цільовому користувачу на основі вподобань інших користувачів. Алгоритм порівнює минулу поведінку цільового користувача з поведінкою інших користувачів. Зокрема, система

присвоює кожному користувачеві вагу, що відображає його схожість з цільовим користувачем, тобто вони є сусідами цільового користувача. Потім вона обирає n -кількість користувачів з найбільшою вагою і прогнозує поведінку цільового користувача (наприклад, прогнозує оцінку до фільму) на основі середньозваженого показника поведінки обраних сусідів. Далі система рекомендує елементи цільовому користувачеві на основі цього прогнозу. Принцип полягає в тому, що якщо цільовий користувач поведився подібно до цієї групи в минулому, він поводитиметься подібним чином і надалі. Функції подібності на основі користувача обчислюються між рядками в матриці «користувач-предмет».

Підхід на основі предмету рекомендує нові предмети цільовому користувачеві на основі його поведінки з подібними предметами. Варто зазначити, що, порівнюючи елементи, колаборативна система порівнює не їхні характеристики, а те, як користувачі взаємодіють з цими елементами. Наприклад, у системі рекомендацій фільмів алгоритм може визначити схожі фільми на основі кореляції між усіма оцінками користувачів для кожного фільму (з поправкою на середню оцінку кожного користувача). Потім система порекомендує новий фільм цільовому користувачеві на основі кореляції оцінок. Тобто, якщо цільовий користувач високо оцінив фільми А і В, але не дивився фільм С, а інші користувачі, які високо оцінили перші два фільми, також високо оцінили фільм С, система порекомендує цільовому користувачеві фільм С. Таким чином, фільтрація на основі елементів обчислює схожість елементів на основі поведінки користувачів. Функції схожості на основі елементів обчислюються між стовпчиками матриці користувач-предмет.

Загалом, одним з найбільших недоліків колаборативної фільтрації на основі пам'яті є те, що її нелегко масштабувати: генерація нової рекомендації може займати надзвичайно багато часу у великих системах. Для систем з мільйонами користувачів і мільйонами елементів крок пошуку найближчих сусідів може стати нерозв'язним, якщо не буде ретельно спроектований

(алгоритм KNN має складність $O(ndk)$ [6], де n - кількість користувачів, d - кількість елементів і k - кількість розглянутих сусідів). Для того, щоб зробити обчислення для великих систем більш зручними, можна скористатися перевагами розрідженості матриці взаємодії при розробці нашого алгоритму або використовувати наближені методи найближчих сусідів (ANN).

В CF на основі моделі передбачається певна латентна модель взаємодії. Така модель навчена реконструювати значення взаємодії користувача і об'єкта на основі власного представлення користувачів та об'єктів. На її основі є можливим створення нових пропозицій. Через те, що передбачається досить вільна модель взаємодії між користувачем та об'єктом, цей підхід теоретично має більшу похибку, але меншу дисперсію [7].

Для реалізації цього підходу найпопулярнішим методами є алгоритми матричної факторизації [8]. Матрична факторизація припускає, що для кожного користувача u та товару i існують латентні векторні представлення $p_u, q_i \in R^f$, тобто r_{ui} можна однозначно виразити - тобто «факторизувати» - через p_u та q_i .

Даний алгоритм факторизує матрицю оцінювання користувач-предмет як добуток двох матриць меншої розмірності, перша з яких має рядок для кожного користувача, а друга - стовпчик для кожного товару. Рядок або стовпець, пов'язані з конкретним користувачем або товаром, називаються латентними факторами. Прогнозовані рейтинги можуть бути розраховані за формулою:

$$\tilde{R} = HW, \quad (1.1)$$

де $\tilde{R} \in \mathbb{R}^{users \times items}$ - матриця оцінювання користувач-предмет, $H \in \mathbb{R}^{users \times latent \ factors}$ містить латентні фактори користувача, а $W \in \mathbb{R}^{latent \ factors \times items}$ - це латентні фактори предметів.

Прогнозований рейтинг, який користувач u надасть предмету i розраховується за наступною формулою:

$$\tilde{r} = \sum_{f=0}^{n \ factors} H_{u,f} W_{f,i} \quad (1.2)$$

Виразність моделі можна налаштувати змінивши кількість латентних факторів. Було продемонстровано [9], що матрична факторизація з одним латентним фактором рекомендує товари з найбільшою кількістю взаємодій без будь-якої персоналізації. Збільшення кількості латентних факторів покращує персоналізацію, а отже, і якість рекомендацій, доки кількість факторів не стане занадто великою, і тоді модель почне перенавчатись, а якість рекомендацій знизиться.

Описаний алгоритм був вперше запропонований Саймоном Фанком [10] і у контексті рекомендаційних систем зазвичай називається SVD (англ. Singular Value Decomposition), проте він не є строго математичним методом SVD, а скоріше наближеним способом обчислення наближення низького рангу матриці шляхом мінімізації квадратичної втрати помилки. Декілька бібліотек на мові Python [11], зокрема бібліотека Surprise, надає чудову реалізацію цього методу.

Великою перевагою колаборативної фільтрації є те, що такий підхід не потребує інформації про користувачів або об'єкти, що є більш привабливим для користувачів, яке переймаються через конфіденційність своїх даних. Також, цей підхід покращується зі збільшенням активності користувача: чим більше він взаємодіє з об'єктами, тим більш актуальнішими і точнішими стають рекомендації.

На противагу цьому, великою проблемою CF є так звана проблема холодного старту (англ. cold start problem): створення рекомендації для нових користувачів або рекомендувати є майже неможливим, а багато користувачів або об'єктів мають занадто мало взаємодій, щоб їх можна було ефективно обробляти [12]. Усунути цю проблему можна, наприклад, створенням рекомендацій випадкових предметів (випадкова стратегія) або рекомендувати популярні предмети новим користувачам або нові предмети найактивнішим користувачам (стратегія максимального очікування), рекомендувати набір різних предметів новим користувачам або новий предмет множині різних

користувачі (дослідницька стратегія), або використати інший підхід на початкових етапах користувача чи предмета.

1.3 Контент фільтрація

Контент фільтрація (англ. content-based filtering) – це система рекомендацій, яка використовує властивості елементів, щоб рекомендувати схожі елементи, які можуть зацікавити користувача [13]. Наприклад, якщо користувачеві сподобався фільм, то система буде рекомендувати схожі фільми базуючись на жанрі або описі.

Контент фільтрація ґрунтується на описі користувача або предмету через їхні метадані. Кожен об'єкт представлений набором релевантних характеристик, таких як наприклад теги за жанром для фільмів. Профіль користувача формується на основі доступної інформації, наприклад, статі, віку або попередньої активності.

Загальні кроки у підході контент фільтрації полягають у вилученні ознак, векторизації тексту, обчислення показника схожості і рекомендації n-кількості предметів (рис. 1.2).

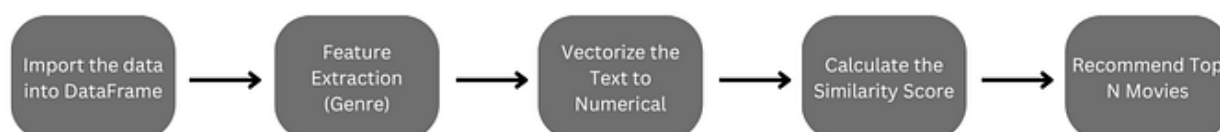


Рис. 1.2. Загальні кроки контент фільтрації

CF системи зображають предмети і користувачів як елементи у векторному просторі. Предмети конвертуються у вектори використовуючи їхні метадані, такі як опис або внутрішні характеристики. Наприклад, при побудові рекомендаційної системи фільмів створюються профілі для кожного фільму з використанням таких метаданих як жанри або опис фільму. Приналежність фільму до певної категорії можна представити за допомогою булевих значень, де 1 вказує на присутність фільму в цій категорії, а 0 - на його відсутність. За

допомогою цієї системи ми потенційно можемо представити невелику кількість фільмів відповідно до жанру. Таблиця 1.1 зображає представлення найпростішого методу векторизації для контент фільтрації – бінарну матрицю характеристик (англ. Binary Feature Matrix).

Таблиця 1.1

Бінарна матриця характеристик

	Пригоди	Анімація	Комедія
«Toy Story»	0	1	1
«Jumanji»	1	0	0
«Lost in the Pacific»	1	0	0
«Waiting to Exhale»	0	0	1

Популярним та якісним методом є TF-IDF, який є розширенням моделі «тобра слів» (англ. Bag of Words) [14].

TF-IDF (англ. Term Frequency – Inverse Document Frequency) – це статистичний показник, що використовується для оцінки важливості слів у контексті документа, що є частиною колекції документів чи корпусу. Векторизація TF-IDF дає числове представлення слів, яке повністю залежить від характеру та кількості документів, що розглядаються [15]. Цей метод складається з визначення частоти терміну (TF – term frequency) та оберненої частоти документа (IDF – inverse document frequency) – інверсія частоти, з якою слово зустрічається в документах колекції. IDF зменшує вагу шикоровживаних слів.

Частота терміну (TF) – це відношення числа входжень обраного слова t до загальної кількості слів в документі d .

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1.3)$$

де $f_{t,d}$ – це число входжень слова в документ, а в знаменнику міститься загальна кількість слів в документі.

Обернена частота документа (IDF) використовується для розрахунку ваги слова, тобто наскільки часто або рідко воно вживається. Часто вживані слова мають меншу вагу ніж унікальні. Цей показник представлений логарифмічно масштабованою зворотною часткою документів, що містять слово, отриманої шляхом ділення загальної кількості документів на кількість документів, що містять термін, а потім взяття логарифма від цього результату:

$$idf(t, D) = \log \frac{1+N}{1+|\{d \in D : t \in d\}|} \quad (1.4)$$

де N – загальна кількість документів в колекції, а $|\{d \in D : t \in d\}|$ – кількість документів, де зустрічається слова t ; якщо слово не в колекції, то це може призвести до ділення на 0, тому зазвичай до чисельника та знаменника додають 1, щоб цьому запобігти.

Сам показник TF-IDF розраховується через множення частоти терміну на обернену частоту документа:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (1.5)$$

Більшу вагу TF-IDF отримають слова з високою частотою появи в межах документа та низькою частотою вживання в інших документах колекції.

Для кращого розуміння варто розглянути наступний приклад. Наявні два речення:

- Речення А: The car is driven on the road.
- Речення В: The truck is driven on the highway.

В цьому прикладі кожне з речень є окремим документом. Розрахунок TF-IDF для цих двох документів, які представляють колекцію, наведено у таблиці 1.2.

Таблиця 1.2

Розрахунок TF-IDF

Слово	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2)=0$	0	0
Car	1/7	0	$\log(2/1)=0.3$	0.043	0
Truck	0	1/7	$\log(2/1)=0.3$	0	0.043
Is	1/7	1/7	$\log(2/2)=0$	0	0
Driven	1/7	1/7	$\log(2/2)=0$	0	0
On	1/7	1/7	$\log(2/2)=0$	0	0
The	1/7	1/7	$\log(2/2)=0$	0	0
Road	1/7	0	$\log(2/1)=0.3$	0.043	0
Highway	0	1/7	$\log(2/1)=0.3$	0	0.043

З таблиці 1.2 видно, що значення TF-IDF часто вживані слова становлять 0, а унікальні слова мають значення відмінне від 0. Такі слова мають більше значення для моделі.

Ще одним методом для представлення слів у векторному просторі є Word2Vec, розроблена компанією Google у 2013 році [16]. Загалом це методика в обробці природної мови (NLP) для отримання векторних зображень слів, яка дозволяє зберігати семантичні відносини між словами, і яка є групою пов'язаних моделей, які являють собою плоскі двошарові нейронні мережі, навчені відтворювати лінгвістичний контекст слів. Word2vec використовує великий текстовий корпус як вхідні дані й створює векторний простір із зазвичай кількома сотнями вимірів. На відміну від TF-IDF, який просто аналізує частоту слів, Word2Vec представляє слова у багатовимірному просторі таким чином, що слова зі схожим значенням розташовуються ближче одне до одного [17].

Модель Word2Vec може бути імплементована з використанням неперервної торби слів (НТС, англ. Continuous Bag of Words) або непервний

пропуск-грам (англ. Continuous Skip-Gram). Обидві моделі намагаються зменшити розмірність даних, але по-різному підходять до цієї проблеми.

НТС використовує навколишні слова, щоб передбачити слово в середині [18]. Дана модель бере всі слова контексту, агрегує їх і використовує отриманий вектор для передбачення цільового слова. Наприклад, у реченні «The cat sat on the mat», якщо ми використовуємо слово «cat» як цільове слово, модель НТС візьме слова «The», «sat», «on», «the», «mat» як контекст і передбачить слово «cat». Ця модель корисна, коли ми маємо невеликий набір даних, і вона працює швидше, ніж модель пропуск-грам. Схема роботи НТС зображена на рисунку 1.3.

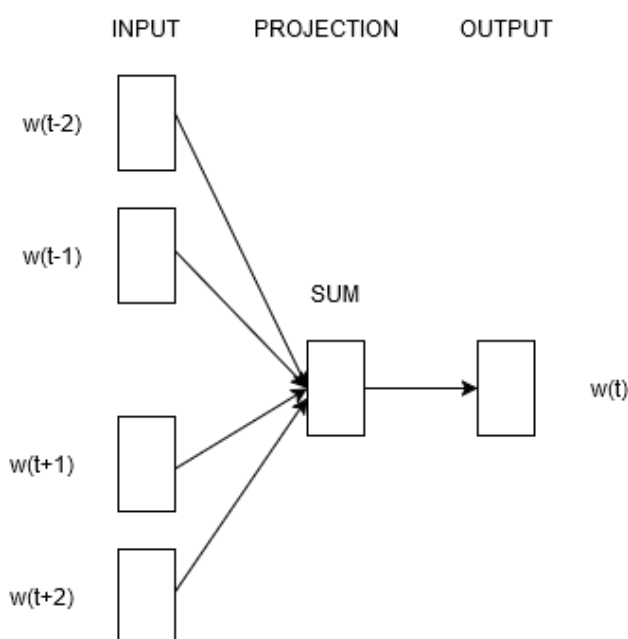


Рис. 1.3. Схема неперервної торби слів

З іншого боку, модель пропуск-грам передбачає навколишній контекст слова, виходячи з цільового слова [19]. Тобто, вона використовує одне слово, щоб передбачити навколишній контекст. Наприклад, якщо ми знову візьмемо речення «The cat sat on the mat», модель візьме слово «cat» як вхідні дані і передбачить «The», «sat», «on», «the», «mat». Така модель добре працює з рідкісними слова, але важкою до обчислення через велику кількість контекстних слів, які треба передбачити. Схема роботи пропуск-грам зображена на рисунку 1.4.

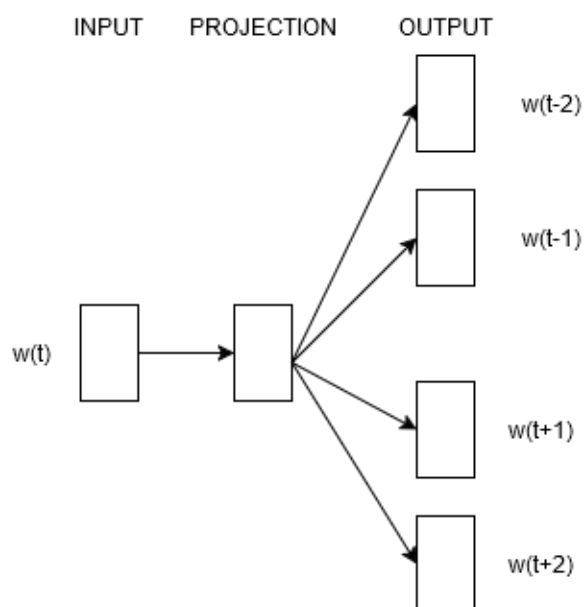


Рис. 1.4. Схема пропуск-грам

Наступним ключовим етапом у побудові рекомендаційних систем є обчислення коефіцієнта схожості між об'єктами або користувачами. Для цього існує багато методів, серед яких є косинус подібності, евклідова відстань і скалярний добуток.

Найпопулярнішим є косинус подібності, який використовується для визначення кута між двома векторами у багатовимірному просторі. Значення цього показника варіюється від -1 до 1: чим ближче воно до 1, тим більш схожими вважаються два об'єкти або користувачі. Ця метрика особливо ефективна у випадках, коли потрібно обробляти високорозмірні простори ознак або текстові дані. Формула для обчислення косинуса подібності виглядає наступним чином:

$$\text{подібність} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (1.6)$$

де A_i та B_i – координати вектору A і B відповідно.

1.3 Гібридні системи

Гібридні системи поєднують найкраще двох світів і є найпопулярнішим варіантом для створення сучасної рекомендаційної системи. Загалом було класифіковано 6 підходів до створення гібридної системи [20]. Проте, варто

зазначити, що суть гібридних систем полягає у поєднанні різних алгоритмів різними способами, тому для більшості проєктів гібридні системи є унікальними, адже підбудовуються під потреби та основні характеристики наявних даних.

Перший підхід є зважувальним і полягає у використанні декількох підходів та надання оцінки їхнім результатам. Система зважених рекомендацій візьме результати роботи кожної з моделей і об'єднає їх у статичні ваги, вага яких не змінюється протягом навчання і тестування набору даних. Наприклад, об'єднується модель на основі контенту та модель спільної фільтрації, і кожна з них матиме вагу 50% для остаточного прогнозу.

Перевага використання зваженого гібридного підходу полягає в тому, що відбувається інтеграція кількох моделей для лінійної підтримки набору даних у процесі рекомендацій. Загальна схема роботи зважувального підходу зображена на рисунку 1.4.

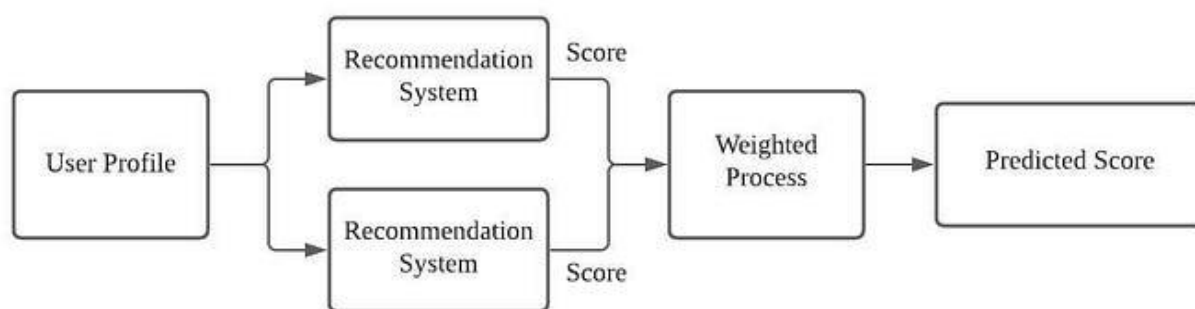


Рис. 1.5. Схема зважувального підходу

Другий вид гібридної системи має назву перемикання. Для створення рекомендацій обирається один з алгоритмів. Такий підхід вводить додатковий рівень на рекомендаційній моделі, який обирає відповідну модель для використання. Він є чутливим до сильних та слабких сторін складової системи рекомендацій. Схема роботи гібридної системи перемикання зображена на рисунку 1.5.

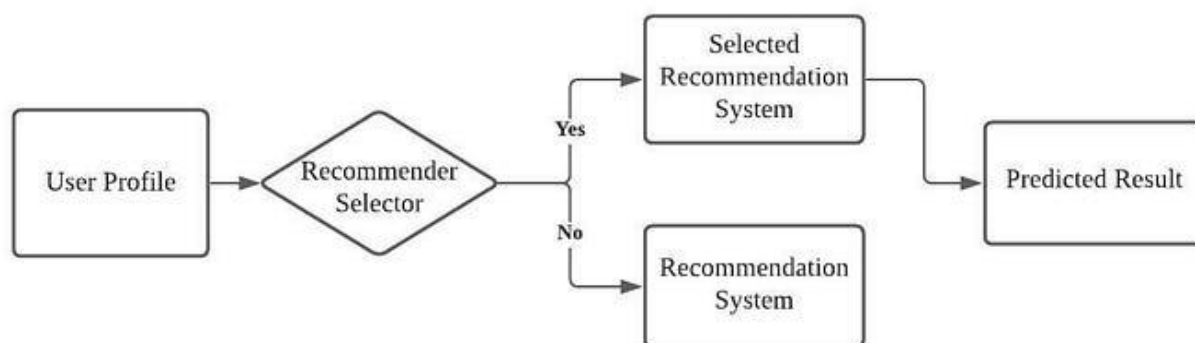


Рис. 1.6. Схема гібридної системи перемикавання

Змішаний підхід є досить популярним вибором у контексті гібридних систем рекомендацій. Його особливість полягає у використанні профіля користувача та його особливостей для створення різних наборів даних. Далі система рекомендацій відповідно вводить ці набори у моделі рекомендацій і комбінує прогнози для отримання рекомендацій. Перевага змішаного підходу полягає у тому, що така система може видавати велику кількість рекомендацій одночасно, а через використання часткового набору даних у відповідній моделі, продуктивність такої системи є кращою. Принцип роботи зображений на схемі на рисунку 1.6.

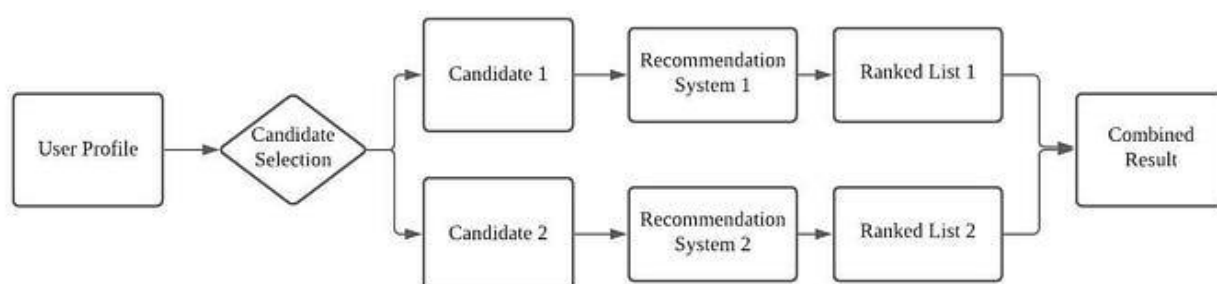


Рис. 1.7. Схема змішаного підходу

Наступним підходом є комбінація характеристик. Принцип роботи такого підходу полягає у додаванні до системи віртуальної моделі рекомендацій, яка доповнює вхідний набір даних додатковими характеристиками. На відміну від підходів, де результати різних алгоритмів комбінуються на етапі формування рекомендацій, цей метод інтегрує характеристики на етапі моделювання. Схема роботи цього підходу зображена на рисунку 1.7.

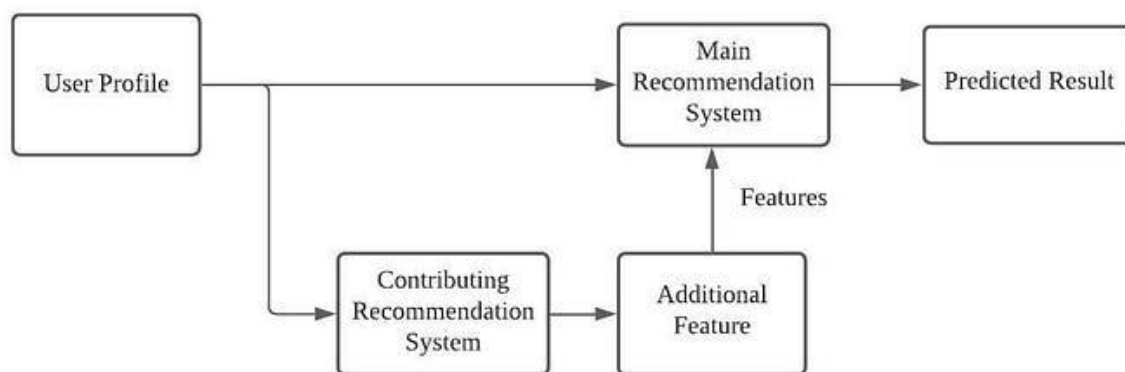


Рис. 1.8. Схема гібридної системи з комбінацією характеристик

Гібридна система з розширенням характеристик полягає у використанні однієї рекомендаційної моделі для створення рейтингу або класифікації профілю користувача чи об'єкту, який далі використовується в основній системі рекомендацій для отримання остаточного результату. У гібридній системі з комбінацією характеристик алгоритми об'єднуються рівноправно, тоді як у системі з розширенням характеристик один з алгоритмів підсилює роботу іншого. Принцип роботи такої системи зображений на рисунку 1.8.



Рис. 1.9. Схема гібридної системи з розширенням характеристик

Каскадний підхід визначає сувору ієрархічну структуру системи рекомендацій. Основна система в такому підході дає первинний результат, який передається у вторинну модель для вирішення деяких неточностей. Такий підхід є ефективним для рівного розподілу балів або вирішенні проблеми з пропущеними даними. Схема каскадного підходу зображена на рисунку 1.9.

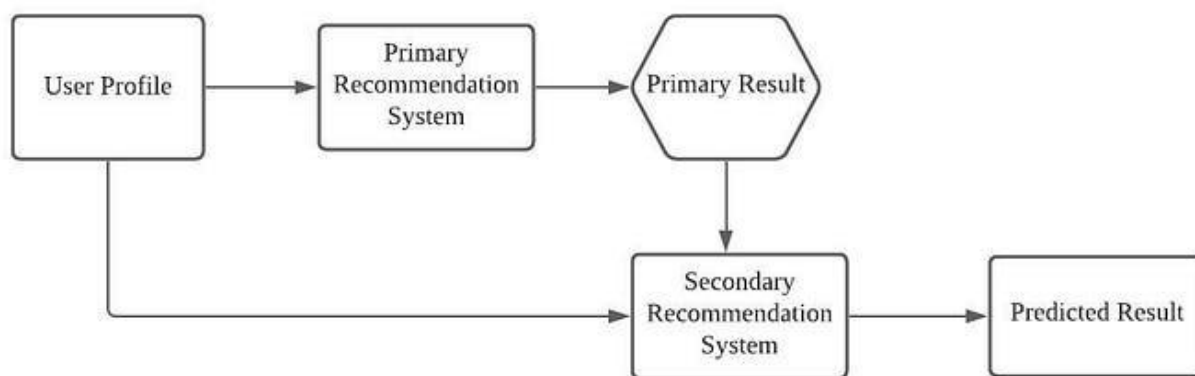


Рис. 1.10. Схема гібридної системи з розширенням характеристик

Висновки до першого розділу

У розділі було подано загальний огляд рекомендаційних систем, описано принципи їхньої роботи, сфери застосування, основні види та ключові проблеми.

Детально розглянуто кожен тип рекомендаційних систем. Для колаборативної фільтрації описано загальний принцип функціонування, проаналізовано особливості підтипів, зокрема було описано підтипи на основі пам'яті та на основі моделі. Також пояснено метод матричної факторизації, який використовується у колаборативній фільтрації на основі моделі.

Було розглянуто схему роботи контент фільтрації, наведено принципи роботи методів TF-IDF та Word2Vec, які використовуються для векторизації тексту, а також детально розглянуто процес розрахунку оцінки схожості.

Окрім цього, представлено й проаналізовано шість підходів для створення гібридних рекомендаційних систем.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНИХ АЛГОРИТМІВ

2.1 Підготовка та аналіз набору даних

Для розробки реалізації рекомендаційних алгоритмів з метою їх подальшого аналізу варто підібрати набір даних, який підійде для розробки будь-якого типу, які були описані у першому розділі.

Основними вимогами до набору даних є:

- наявність опису та тегів у предметів;
- наявність оцінок користувачів.

Остаточний набір даних, який був обраний, це The Movies Dataset з платформи Kaggle. На рисунку 2.1 зображена сторінка набору даних.

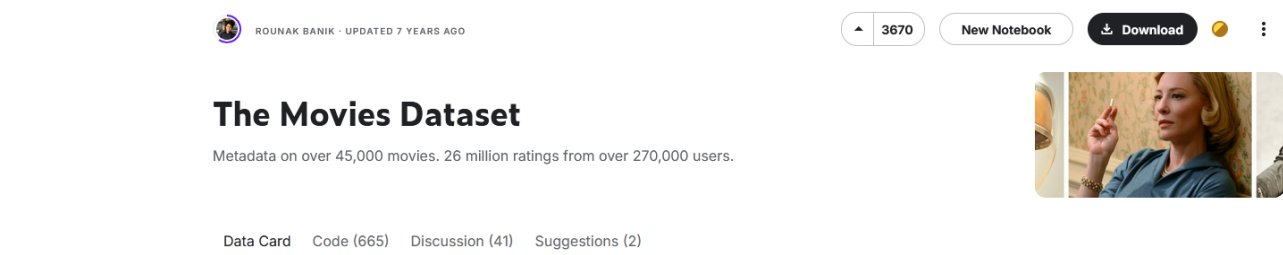


Рис. 2.1. The Movies Dataset на платформі Kaggle

За описом [21] цей набір даних містить 45 000 фільмів, а також 26 мільйонів користувацьких рейтингів, з набору даних від MovieLens, який був випущений у 2017 році. Він містить наступні файли:

- movies_metadata.csv містить інформацію про фільми;
- keywords.csv містить ключові слова сюжетів;
- credits.csv містить інформацію акторів та команду;
- links.csv містить ідентифікатори TMDb та IMDb всіх фільмів;
- links_small.csv містить ідентифікатори TMDb та IMDb підгрупи фільмів;
- ratings_small.csv містить підгрупу з 100 000 рейтингів від 700 користувачів до 9 000 фільмів;
- ratings.csv містить 26 мільйонів рейтингів від 270 000 користувачів до 45 000 фільмів.

З наявних файлів тільки файли з інформацією про фільми (movies_metadata.csv) та інформацією про рейтинги (ratings.csv) є необхідними для розробки рекомендаційних систем для їх подальшого аналізу.

Для успішної реалізації рекомендаційних алгоритмів першим кроком є проведення розвідувального аналізу набору даних та його підготовки. Цей етап дозволить вивчити структуру даних, оцінити їхню якість та вирішити питання, які можливо призведуть до проблем. Під час цього етапу також буде проведена очистка даних та їх підготовка до подальшого використання. Вся робота проводиться з допомогою бібліотеки Pandas, яка дозволяє працювати з наборами даних [22].

Дослідження набору даних почато з файлу, який містить фільми та їхню інформацію. Прочитавши файл і завантаживши його в змінну як об'єкт DataFrame, можливо переглянути загальну кількість записів, що становить 45466, та кількість колонок, що становить 24, а також перші п'ять записів для кращого розуміння. Вони представлені на рисунку 2.2.

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	release_date	revenue	runtime	spoken_languages	status	tagline	title	video	vote_average	vote_count
0	False	[{"id": 10194, "name": "Toy Story Collection", ...}]	30000000	[{"id": 16, "name": "Animation"}, {"id": 35, "name": "Comedy"}]	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy & his five friends happily in his ...	1995-10-30	37354033.0	81.0	[{"iso_639_1": "en", "name": "English"}]	Released	NaH	Toy Story	False	7.7	5415.0
1	False	NaH	65000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Action"}, {"id": 35, "name": "Comedy"}]	NaH	8844	tt0113497	en	Jurassic Park	When siblings Judy and Peter discover an escaped ...	1995-12-15	262797249.0	104.0	[{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "es", "name": "Spanish"}]	Released	Roll the dice and unleash the excitement!	Jurassic Park	False	6.9	2413.0
2	False	[{"id": 119050, "name": "Grumpy Old Men Collect..."}]	0	[{"id": 10248, "name": "Romance"}, {"id": 35, "name": "Comedy"}]	NaH	15602	tt0113228	en	Grumpy Old Men	A family wedding reignites the ancient feud be...	1995-12-22	0.0	101.0	[{"iso_639_1": "en", "name": "English"}]	Released	Still Yelling. Still Fighting. Still Ready for...	Grumpy Old Men	False	6.5	52.0
3	False	NaH	16000000	[{"id": 35, "name": "Comedy"}, {"id": 18, "name": "Drama"}]	NaH	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	1995-12-22	81452156.0	127.0	[{"iso_639_1": "en", "name": "English"}]	Released	Friends are the people who let you be yourself...	Waiting to Exhale	False	6.1	34.0
4	False	[{"id": 96871, "name": "Father of the Bride Col..."}]	0	[{"id": 35, "name": "Comedy"}]	NaH	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	1995-02-10	76578911.0	106.0	[{"iso_639_1": "en", "name": "English"}]	Released	Just When His World Is Back To Normal... He's ...	Father of the Bride Part II	False	5.7	173.0

Рис. 2.2. Перші п'ять записів файлу movies_metadata.csv

Назви всіх колонок представлені на рисунку 2.3.

```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
      'imdb_id', 'original_language', 'original_title', 'overview',
      'popularity', 'poster_path', 'production_companies',
      'production_countries', 'release_date', 'revenue', 'runtime',
      'spoken_languages', 'status', 'tagline', 'title', 'video',
      'vote_average', 'vote_count'],
      dtype='object')
```

Рис. 2.3. Назви колонок

Деякі з цих колонок абсолютно не важливі ні для дослідження набору даних, ні для реалізації рекомендаційної системи, тому було видалено наступні

колонки: budget, adult, belongs_to_collection, homepage, spoken_language, tagline, video, poster_path, status, vote_count, popularity, imdb_id, revenue.

Далі відбулася перевірка на наявність пропущених значень. Результат перевірки зображений на рисунку 2.4.

```
genres      0
id          0
original_language  11
original_title  0
overview    954
production_companies  3
production_countries  3
release_date  87
runtime     263
title       6
vote_average  6
dtype: int64
```

Рис. 2.4. Перевірка на наявність порожніх значень

Оскільки стовпчик overview має опис фільму, що є важливим для розробки рекомендаційної системи на основі контент фільтрації, але число пропущених значень у співвідношенні до всієї кількості фільмів є невеликим, було просто видалено всі пропущені значення з набору даних за допомогою вбудованої функції бібліотеки Pandas dropna().

Дані про жанри, кінокомпанії та країни представлені у форматі JSON, який був перетворений на рядок. Для того, щоб ці дані було простіше використовувати їх було перетворено у зручний для читання текстовий формат, методом зображеним на рисунку 2.5.

```
movies['genres'] = movies['genres'].apply(lambda x: ', '.join([d['name'] for d in eval(x)]) if x else '')
movies['production_companies'] = movies['production_companies'].astype(str)
movies['production_companies'] = movies['production_companies'].apply(lambda x: ', '.join([d['name'] for d in eval(x)]) if x else '')
movies['production_countries'] = movies['production_countries'].apply(lambda x: ', '.join([d['name'] for d in eval(x)]) if x else '')
```

Рис. 2.5. Перетворення даних

Метод apply() застосовую вказану лямбда-функцію до кожного значення стовпця genres. Оскільки значення, які перетворюються, мають сталий шаблон, який схожий на Python-словник, було використано функцію eval(), який перетворює наявні рядки. Далі з кожного словника витягалося лише значення name, які об'єднуються в один рядок, розділений комами. Якщо ж значення, яке

обробляється є порожнім або NaN, то записується прожній рядок. Додатково також була перероблена дата виходу фільму, в результаті чого було записано лише рік (без дня та місяця). Результ проробленої роботи зображений на рисунку 2.6 як 5 випадково вибраних записів.

	genres	id	original_language	original_title	overview	production_companies	production_countries	release_date	runtime	title	vote_average
25357	Documentary	241953	en	Independents	Discover what it really takes to strike out on...			2007	77.0	Independents	0.0
13348	Thriller, Action, Crime	14549	en	Night Train	Two Passengers (Sobieski and Zahn) and the con...	A-Mark Entertainment, Instinctive Film, FilmTiger	Germany, Romania, United States of America	2009	87.0	Night Train	5.4
6095	Romance, Comedy	44968	en	Born Yesterday	A businessman shows up in Washington to lobby ...	Hollywood Pictures		1993	100.0	Born Yesterday	4.8
26432	Action	173793	en	White Tiger	When D.E.A. agent Mike Ryan undertakes the hug...	Keystone Pictures, Den Pictures Inc., Gail For...	Canada, United States of America	1996	93.0	White Tiger	6.0
39200	Crime	73307	en	No Man's Woman	A greedy, scheming woman is found murdered in ...	Republic Pictures (I)	United States of America	1955	70.0	No Man's Woman	4.3

Рис. 2.6. П'ять випадкових записів після форматування даних

Для зручності у подальшій роботі було також перейменовано стовпчик 'id' на 'movieId', стовпчик 'release_date' на 'release_year'. Відредаговані дані було збережено в окремий файл під назвою movies.csv.

Для кращого розуміння розподілу даних було створено декілька діаграм з використанням бібліотки Matplotlib [23] і Seaborn [24], зокрема діаграма розподілу середніх рейтингів фільмів, яка зображена на рисунку 2.7.

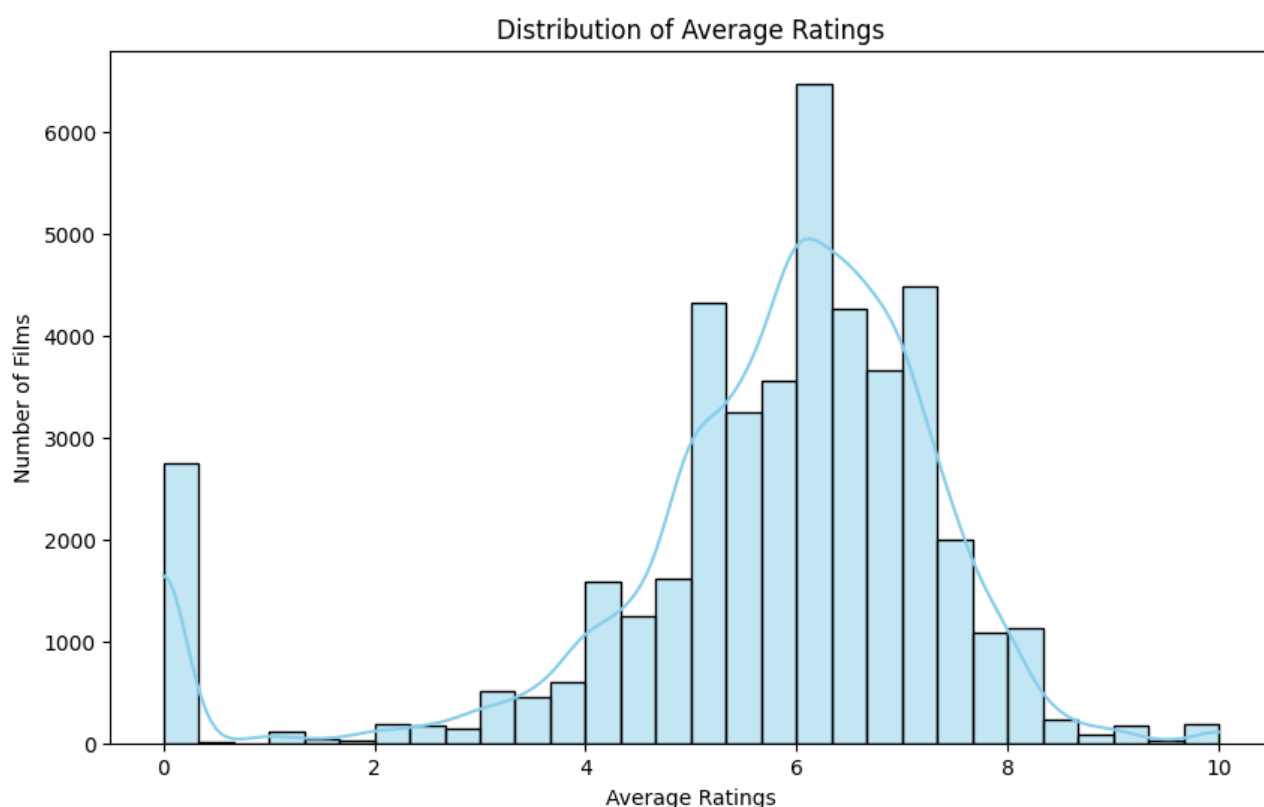


Рис. 2.7. Діаграма середніх рейтингів фільмів

Дана гістограма показує розподіл середніх оцінок фільмів. Стає зрозумілим, що значна кількість фільмів має середню оцінку 6 за шкалою від 1 до 10. Наступною була створена діаграма частоту появи різних жанрів, яка зображена на рисунку 2.8.

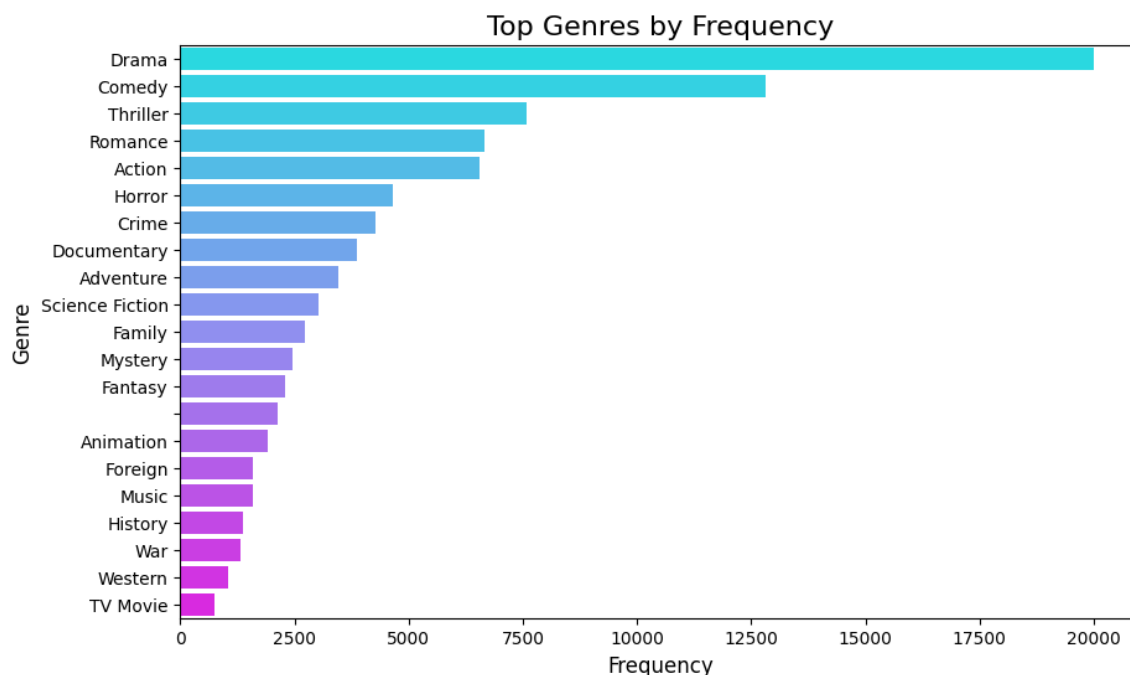


Рис. 2.8. Частота появи жанрів

За даними цієї діаграми найпопулярнішими жанрами є драма, комедія та трилер. Найменш популярними є телевізійні фільми, вестерни та фільми про війну. Далі було проілюстровано найбільш поширені мови у фільмах, що зображено у вигляді секторної діаграми на рисунку 2.9.

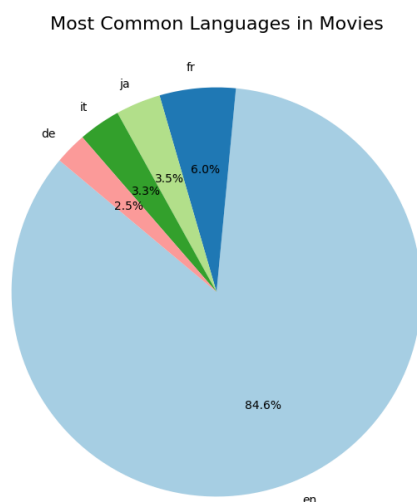


Рис. 2.9. Секторна діаграма мов, які найбільше зустрічаються

Діаграма зображає, що значна частка фільмів знімається англійською мовою (84.6%), на другому місці знаходиться французька мова (лише 6%). Також було створена гістограма для середніх оцінок топ-5 найпопулярніших жанрів, яка зображена на рисунку 2.10.

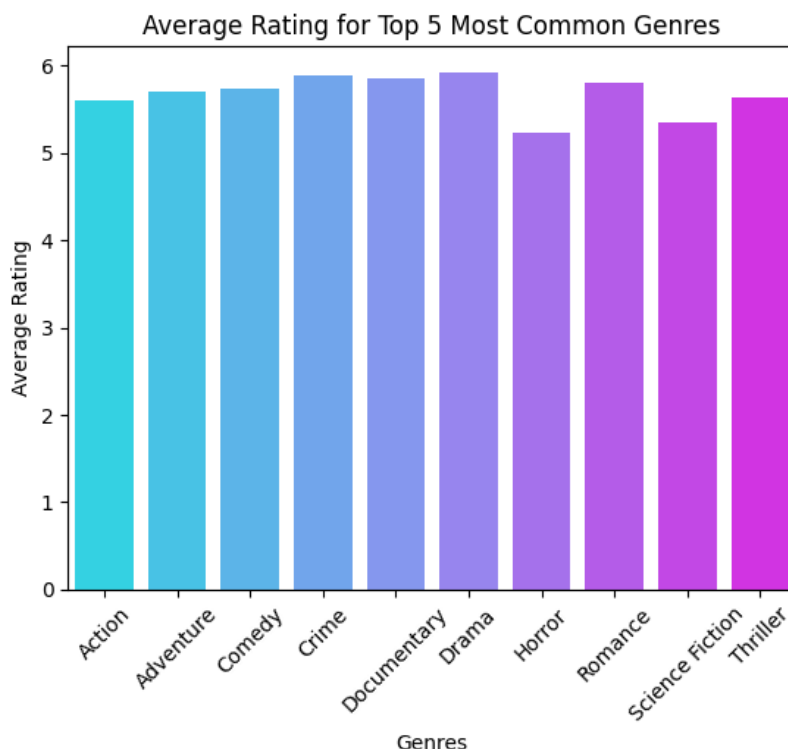


Рис. 2.10. Гістограма середніх оцінок для топ-5 найпопулярніших жанрів

Дана діаграма ілюструє доволі сталу оцінку (у межах від 5 до 6) для найбільш популярних жанрів.

Загальний лістинг форматування та дослідження даних фільмів наведено у Додатку А.

2.2 Реалізація колаборативної фільтрації

Для реалізації рекомендаційної системи з використанням колаборативної фільтрації був використаний алгоритм матричної факторизації SVD з бібліотеки Surprise для Python [25]. Загальна функція для отримання рекомендацій зображена на рисунку 2.11.

```

def cf_top_rec(userId, ratings):
    cf_df = pd.DataFrame()
    movie_id_list = ratings['movieId'].unique().tolist()
    est = []
    movie_name_list = []
    actual_score_list = []
    genre_list = []

    for i in movie_id_list:
        pred = svd.predict(userId, i, verbose=False)
        est.append(pred[3])
        movie_name = movies[movies.movieId == i]['title'].values
        movie_name_list.append(movie_name)
        actual_score = ratings[(ratings.movieId==i)&(ratings.userId==userId)]['rating'].values
        actual_score_list.append(actual_score)
        genre = movies[movies.movieId == i]['genres'].values
        try:
            genre_list.append(genre)
        except:
            genre_list.append('Unavailable')
    cf_df['movieId'] = movie_id_list
    cf_df['estimated_rating'] = est
    cf_df['title'] = movie_name_list
    cf_df['actual_rating'] = actual_score_list
    cf_df['genres'] = genre_list
    cf_df = cf_df.sort_values(['estimated_rating'], ascending=False)
    return cf_df

```

Рис. 2.11. Функція колаборативної фільтрації

Дана функція приймає ідентифікатор користувача, для якого створюються рекомендації, та інформацію про рейтинги. Використовується модель SVD і її функція `predict()` для прогнозування рейтингу, який користувач ймовірно поставить фільму. Цей рейтинг записується до списку, який потім утворює стовпчик `'estimated_rating'`. Утворюється новий датафрейм з загальною інформацією про рекомендовані фільми. Перші 10 рекомендацій для користувача з ідентифікатором 20 зображені на рисунку 2.12.

	movieId	estimated_rating	title	actual_rating	genres
855	215	4.789255	[Saw II]	[]	['Horror']
64	318	4.725812	[The Million Dollar Hotel]	[]	['Drama', 'Thriller']
2	858	4.683654	[Sleepless in Seattle]	[4.5]	['Comedy', 'Drama', 'Romance']
66	5618	4.671043	[Cousin, Cousine]	[4.5]	['Romance', 'Comedy']
6	2959	4.663812	[License to Wed]	[4.0]	['Comedy']
54	296	4.654757	[Terminator 3: Rise of the Machines]	[4.5]	['Action', 'Thriller', 'Science Fiction']
1081	127098	4.634038	[Extraordinary Illusions]	[]	['Comedy']
40	1732	4.614485	[The Prisoner of Zenda]	[4.5]	['Action', 'Adventure', 'Comedy']
65	4993	4.606657	[5 Card Stud]	[3.5]	['Action', 'Western', 'Thriller']
3234	26644	4.589991	[Photographing Fairies]	[]	['Mystery', 'Fantasy', 'Drama']

Рис. 2.12. Рекомендації колаборативної фільтрації

Максимальна можлива оцінка у фільму становить 5. Загалом надані рекомендації є різноманітними за жанрами. Столпчик 'actual_rating' має переважно порожні значення, що є нормальним і свідчить про те, що скоріше за все рекомендований фільм є новинкою для користувача. Загальний лістинг рекомендаційної системи колаборативної фільтрації наведено у додатку Б.

2.3 Реалізація контент фільтрації

Реалізація контент фільтрації була реалізована окремо для жанрів і для описів фільмів. Для обох рекомендаційних систем була використана бібліотека sklearn, зокрема метод векторизації TF-IDF [26]. Для початку було ініціалізовано метод TfidfVectorizer(), далі було обчислено матрицю TF-IDF, для якої столпчик genres розбивається на уніграми та біграми. Ця матриця є розрідженою, тому для зручності роботи її було перетворено на щільну. Вона представлена на рисунку 2.13.

	action	action adventure	action animation	action comedy	action crime	action documentary	action drama	action family	action fantasy	action foreign	western foreign	western history	western horror	western music	western mystery	western romance	western science	western thriller	western tv
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
44420	0.288485	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44421	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44422	0.366273	0.0	0.0	0.0	0.0	0.0	0.627937	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44423	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44424	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

44425 rows x 401 columns

Рис. 2.13. Матриця TF-IDF

Дана матриця має розмір 44425 на 401. Далі був розрахований косинус подібності отриманої матриці, результат обчислення наведений на рисунку 2.14.

```
array([[1.          , 0.12577275, 0.07598416, 0.07503619, 0.22747319],
       [0.12577275, 1.          , 0.          , 0.          , 0.          ],
       [0.07598416, 0.          , 1.          , 0.29371713, 0.33403568],
       [0.07503619, 0.          , 0.29371713, 1.          , 0.32986828],
       [0.22747319, 0.          , 0.33403568, 0.32986828, 1.          ]])
```

Рис. 2.14. Косинус подібності

Наступною була створена функція для створення рекомендацій, яка зображена на рисунку 2.15.

```

def genre_rec(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:]
    movie_indices = [i[0] for i in sim_scores]
    cb_df = pd.DataFrame(sim_scores)
    cb_df['title'] = titles.iloc[movie_indices].values
    cb_df['movieId'] = movie_id.iloc[movie_indices].values
    cb_df = cb_df.rename(columns={0: "movie_index", 1: "similarity_score"})
    genre_list = []
    for i in cb_df['movieId'].values:
        genre = movies[movies.movieId==i]['genres'].values
        try:
            genre_list.append(genre)
        except:
            genre_list.append('Unavailable')
    cb_df['genres'] = genre_list
    return cb_df

```

Рис. 2.15. Функція для створення рекомендацій

На початку обчислюється косинус подібності між вхідним фільмом і всіма іншими в матриці, яка містить попередньо обчислені значення косинусу подібності між векторами жанрів фільмів, з використанням індексу вхідного фільму для вибору відповідного рядка. Потім кожен елемент рядка перетворюється на пару (індекс фільму, значення подібності) за допомогою методу `enumerate()`. Список цих пар сортується за значеннями подібності в порядку спадання, щоб фільми з найбільшою подібністю йшли першими. Після цього видаляється перший елемент списку, оскільки він завжди відповідає самому вхідному фільму (подібність із собою дорівнює 1). Це дозволяє залишити тільки фільми, які є справді схожими. Інформація про них записується у новий датафрейм. Перші 20 рекомендацій за фільмом «Harry Potter and the Prisoner of Azkaban» зображені на рисунку 2.16.

movie_index	similarity_score	title	movielid	genres	
0	4742	1.000000	Harry Potter and the Philosopher's Stone	671	[['Adventure', 'Fantasy', 'Family']]
1	5381	1.000000	Clash of the Titans	10323	[['Adventure', 'Fantasy', 'Family']]
2	5653	1.000000	Harry Potter and the Chamber of Secrets	672	[['Adventure', 'Fantasy', 'Family']]
3	6073	1.000000	Journey to the Center of the Earth	11571	[['Adventure', 'Fantasy', 'Family']]
4	6982	1.000000	Peter Pan	10601	[['Adventure', 'Fantasy', 'Family']]
5	7696	1.000000	Harry Potter and the Prisoner of Azkaban	673	[['Adventure', 'Fantasy', 'Family']]
6	10519	1.000000	Harry Potter and the Goblet of Fire	674	[['Adventure', 'Fantasy', 'Family']]
7	10859	1.000000	Bröderna Lejonhjärta	11291	[['Adventure', 'Fantasy', 'Family']]
8	11942	1.000000	Jack the Giant Killer	25360	[['Adventure', 'Fantasy', 'Family']]
9	13841	1.000000	Harry Potter and the Half-Blood Prince	767	[['Adventure', 'Fantasy', 'Family']]
10	14775	1.000000	Percy Jackson & the Olympians: The Lightning T...	32657	[['Adventure', 'Fantasy', 'Family']]
11	16048	1.000000	Harry Potter and the Deathly Hallows: Part 1	12444	[['Adventure', 'Fantasy', 'Family']]
12	23252	1.000000	Neverland	106938	[['Adventure', 'Fantasy', 'Family']]
13	27433	1.000000	Cesta do pravěku	8779	[['Adventure', 'Fantasy', 'Family']]
14	37464	1.000000	Starik Khottabych	71103	[['Adventure', 'Fantasy', 'Family']]
15	32041	0.821258	Arthur 3: la guerre des deux mondes	42949	[['Animation', 'Adventure', 'Fantasy', 'Family']]
16	36688	0.821258	Pohádka o Honzíkovi a Mařence	51902	[['Animation', 'Adventure', 'Fantasy', 'Family']]
17	8530	0.798654	The Wonderful World of the Brothers Grimm	28367	[['Fantasy', 'Family']]
18	9631	0.798654	Alice in Wonderland	30923	[['Fantasy', 'Family']]
19	18745	0.798654	Don't Eat the Pictures: Sesame Street at the M...	319656	[['Fantasy', 'Family']]

Рис. 2.16. Рекомендація контент фільтрації

Отримані рекомендації є дуже схожими на вхідний фільм. Лістинг рекомендаційного алгоритму контент фільтрації наведений у додатку В.

Наступним був реалізований підхід на основі TF-IDF та Word2Vec. Для початку був форматований опис фільмів, а саме переведено текст у нижній регістр, прибрано загальні англійські слова («the», «and», «is») та прибрано пунктуацію. Отриманий результат було записано у стовпчик cleaned і зображено на рисунку 2.17.

genres	movielid	original_language	original_title	overview	production_companies	production_countries	release_year	runtime	title	vote_average	cleaned
['Animation', 'Comedy', 'Family']	862	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	Pixar Animation Studios	United States of America	1995	81.0	Toy Story	7.7	led woody andy s toys live happily room andy s...
['Adventure', 'Fantasy', 'Family']	8844	en	Jumanji	When siblings Judy and Peter discover an encha...	TriStar Pictures, Teitler Film, Interscope Com...	United States of America	1995	104.0	Jumanji	6.9	siblings judy peter discover enchanted board 9...
['Romance', 'Comedy']	15602	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	Warner Bros., Lancaster Gate	United States of America	1995	101.0	Grumpier Old Men	6.5	family wedding reignites ancient feud next doo...
['Comedy', 'Drama', 'Romance']	31357	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	Twentieth Century Fox Film Corporation	United States of America	1995	127.0	Waiting to Exhale	6.1	cheated on mistreated stepped on women holding... george banks

Рис. 2.17. Відформатований опис

Для кращого розуміння даних було створено гістограму 20 біграм, які найчастіше зустрічаються в описах наявних фільмів. Діаграма зображена на рисунку 2.18. Також було створено схожу гістограму для триграм, яка зображена на рисунку 2.19.

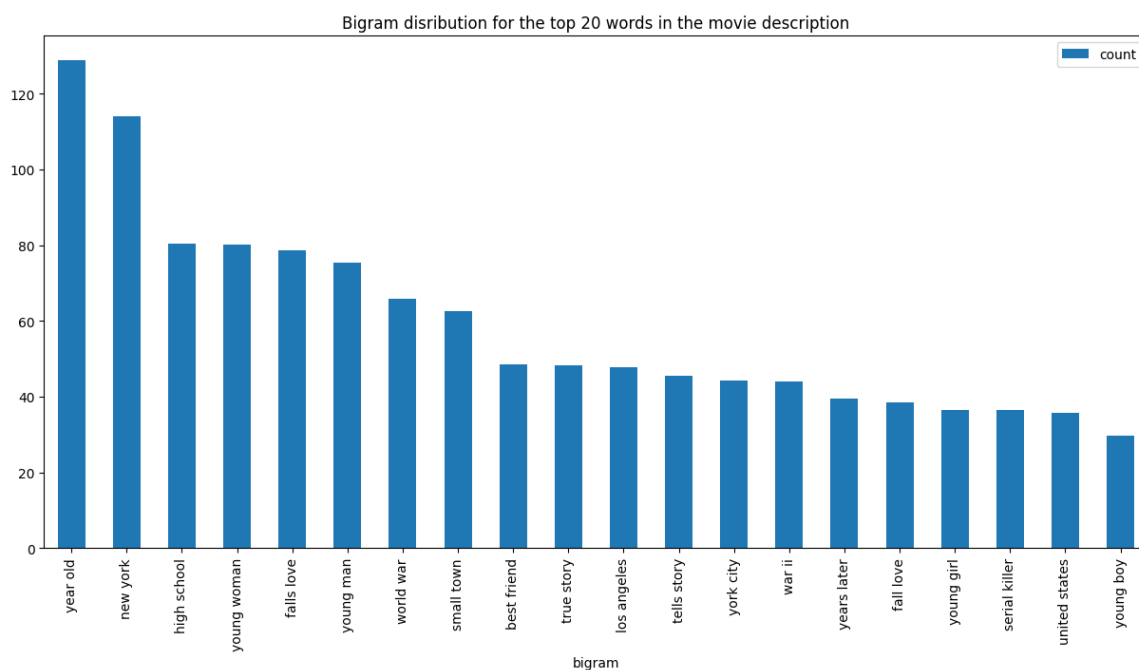


Рис. 2.18. Біграми, які найчастіше зустрічаються

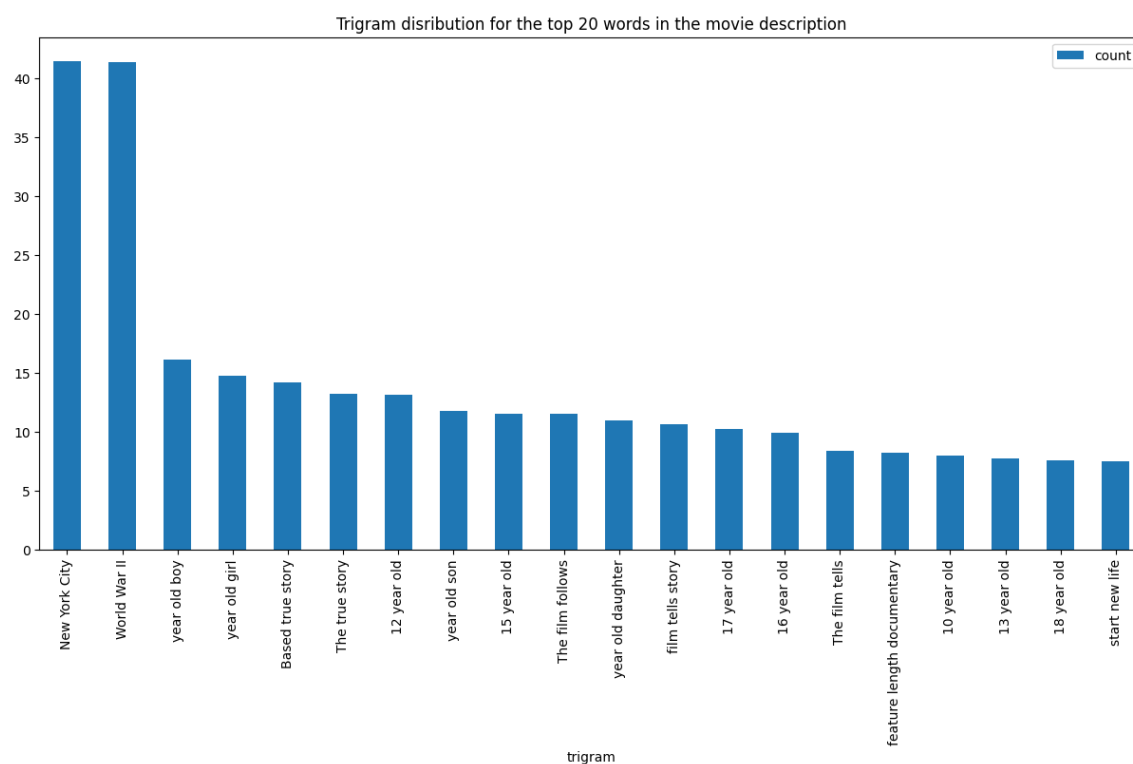


Рис. 2.19. Триграми, які найчастіше зустрічаються

З гістограм видно, що найпопулярнішими біграмами є «year old» і «new york», а серед триграм це «new york city» і «world war II». Для реалізації методу Word2Vec [27] було використано бібліотеку genism [28], а також вже попередньо навчену модель від Google. Напочатку було створено власний корпус, який містить слова, які зустрічаються в очищеному описі фільмів. Попередньо навчені

вектори з моделі від Google були адаптовані під наявний корпус шляхом навчання моделі, що зробить результати більш спеціалізованими. Лістинг методу зображений на рисунку 2.20.

```
google_model = Word2Vec(vector_size = 300, window=5, min_count = 2, workers = 2)
google_model.build_vocab(corpus)

google_model.wv.load_word2vec_format(EMBEDDING_FILE, binary=True)

google_model.train(corpus, total_examples=google_model.corpus_count, epochs = 5)
```

Рис. 2.20. Навчання моделі на власному корпусі

Далі для кожного векторного представлення слів була розрахована вага TF-IDF, яка потім була поєднана з результатами Word2Vec. Результатом є список векторів, кожен з яких представляє один текст у багатовимірному просторі. Наступним кроком є розрахунок косинусу подібності для визначення схожих значень. Функція, зображена на рисунку 2.21, повертає список рекомендацій на основі обраного фільму.

```
def desc_rec(title):

    movies_df = movies[['title', 'overview']]
    indices = pd.Series(movies.index, index = movies['title']).drop_duplicates()

    idx = indices[title]
    sim_scores = list(enumerate(cosine_similarities[idx]))
    sim_scores = sorted(sim_scores, key = lambda x: x[1], reverse = True)
    sim_scores = sim_scores[1:]
    movie_indices = [i[0] for i in sim_scores]
    cb_desc_df = pd.DataFrame(sim_scores)
    cb_desc_df['title'] = titles.iloc[movie_indices].values
    cb_desc_df['movieId'] = movie_id.iloc[movie_indices].values
    cb_desc_df = cb_desc_df.rename(columns={0: "movie_index", 1: "similarity_score"})
    genre_list_desc = []
    for i in cb_desc_df['movieId'].values:
        genre = movies[movies.movieId==i]['genres'].values
        try:
            genre_list_desc.append(genre)
        except:
            genre_list_desc.append('Unavailable')
    cb_desc_df['genres'] = genre_list_desc
    return cb_desc_df
```

Рис. 2.21. Функція для створення списку рекомендацій

Рекомендації були створені за фільмом «Harry Potter and the Prisoner of Azkaban» і зображені на рисунку 2.22.

	movie_index	similarity_score	title	movieid	genres
0	18185	0.977138	Intruders	63207	[['Horror', 'Thriller', 'Drama']]
1	19499	0.975216	Astérix & Obélix - Au service de Sa Majesté	99770	[['Family', 'Adventure', 'Comedy']]
2	32266	0.973830	ストリートファイターZERO	17433	[['Action', 'Animation']]
3	13246	0.973410	The Dungeonmaster	27244	[['Horror', 'Fantasy', 'Thriller', 'Science Fi...
4	15737	0.973291	Infestation	22476	[['Action', 'Comedy', 'Horror', 'Science Ficti...
5	26057	0.971813	The Land Before Time VII: The Stone of Cold Fire	24102	[['Adventure', 'Fantasy', 'Animation', 'Scienc...
6	22571	0.971474	El canibal	84362	[['Horror']]
7	5424	0.971089	Ernest Scared Stupid	32685	[['Horror', 'Comedy', 'Family']]
8	19995	0.970739	るろうに剣心	127533	[['Action', 'Drama', 'History']]
9	14861	0.970567	ルパン三世 sweet lost night ~魔法のランプは悪夢の予感~	59719	[['Action', 'Animation', 'Comedy', 'TV Movie']]
10	41813	0.970553	聖闘士星矢 真紅の少年伝説	66105	[['Action', 'Animation', 'Fantasy']]

Рис. 2.22. Список рекомендацій

Повний лістинг контент фільтрації з використанням TF-IDF і Word2Vec зображений у додатку Г.

2.4 Реалізація гібридної системи

Для створення гібридної системи рекомендацій використовувався каскадний підхід. Загалом функція приймає ідентифікатор користувача, назву фільму та набір даних з рейтингами. Спочатку через контент фільтрацію розраховується список рекомендацій, далі використовується колаборативна фільтрація для визначення оцінок, які може поставити обраний користувач. Загальний лістинг гібридного підходу наведений у додатку Д. Список рекомендацій для користувача з ідентифікатором 1 для фільму «Harry Potter and the Prisoner of Azkaban» зображений на рисунку 2.23.

	movieid	similarity_score	title	estimated_rating	actual_rating
3	6073	1.000000	Journey to the Center of the Earth	4.220434	□
0	4742	1.000000	Harry Potter and the Philosopher's Stone	3.863778	□
1	5381	1.000000	Clash of the Titans	3.863778	□
4	6982	1.000000	Peter Pan	3.863778	□
5	7696	1.000000	Harry Potter and the Prisoner of Azkaban	3.863778	□
6	10519	1.000000	Harry Potter and the Goblet of Fire	3.863778	□
7	10859	1.000000	Bröderna Lejonhjärta	3.863778	□
8	11942	1.000000	Jack the Giant Killer	3.863778	□
9	13841	1.000000	Harry Potter and the Half-Blood Prince	3.863778	□
10	14775	1.000000	Percy Jackson & the Olympians: The Lightning T...	3.863778	□
11	16048	1.000000	Harry Potter and the Deathly Hallows: Part 1	3.863778	□
12	23252	1.000000	Neverland	3.863778	□
13	27433	1.000000	Cesta do pravěku	3.863778	□
14	37464	1.000000	Starik Khottabych	3.863778	□
2	5653	1.000000	Harry Potter and the Chamber of Secrets	3.787487	□
15	32041	0.821258	Arthur 3: la guerre des deux mondes	3.863778	□
16	36688	0.821258	Pohádka o Honzíkovi a Mařence	3.863778	□

Рис. 2.23. Список рекомендацій

Висновки до другого розділу

У цьому розділі було обрано набір даних з інформацією про фільми від MovieLens, проведено його обробку та розвідувальний аналіз. Було реалізовано чотири рекомендаційні системи. Зокрема, впроваджено колаборативну фільтрацію із застосуванням методу SVD, детально описано принцип роботи імплементованого алгоритму та представлено результати його застосування у формі таблиці рекомендацій, яка містить інформацією про фільми.

Для контент фільтрації було реалізовано дві системи: перша на основі аналізу жанрів із використанням методу TF-IDF, іншу — на основі аналізу описів, для чого було поєднано TF-IDF із Word2Vec. У роботі наведено принципи функціонування обох підходів та проаналізовано результати їх застосування.

Гібридну рекомендаційну систему розроблено з використанням каскадного підходу. На першому етапі застосовувалася контентна фільтрація за жанрами, після чого на отриманих результатах проводилася колаборативна фільтрація. У роботі наведено отримані рекомендації, що демонструють результат використання підходу.

РОЗДІЛ 3 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

3.1 Визначення критеріїв оцінювання

Рекомендаційні системи є досить складними для оцінювання через певну суб'єктивність. Також точність оцінки рекомендаційного алгоритму і загалом роботи всієї системи залежить від набору даних та їх точність. Наприклад, якщо наявні багато користувачів з кардинально різними смаками (йому подобаються і мультфільми, і документальні фільми про війну), то це може негативно повпливати на систему.

Перехресна валідація являється статистичним методом оцінювання продуктивності моделі машинного навчання. Основна ідея цього методу полягає у розбитті вхідного набору даних на частини для навчання та тестування, щоб отримати надійні оцінки якості моделі. Найбільш популярним підходом є k -кратна перехресна валідація. В цьому підході набір даних ділиться на k рівних частин, модель навчається на $k-1$ частинах і тестується на решті. Цей повторюється k разів.

Стандартним методом [29] для оцінки моделі також є визначення показника кореня середньоквадратичної похибки (RMSE, англ. root mean square error) [30]. Цей показник обчислюється за наступною формулою:

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}, \quad (3.1)$$

де $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ – прогнозовані значення; $y_1, y_2, \dots, y_n$ – справжні значення; n – кількість значень.

Ділення на n під квадратним коренем у RMSE дозволяє оцінити стандартне відхилення похибки для окремого спостереження, замість того, щоб представляти загальну похибку. Ця нормалізація гарантує, що похибка залишатиметься незмінною зі збільшенням набору даних, стаючи більш точною зі збільшенням кількості спостережень. RMSE також являється більш інтуїтивно зрозумілою, адже предствалена у тих самих одиницях, що і прогнозовані значення.

Середньоквадратична похибка (англ. mean square error, MSE) натомість використовує квадратні одиниці, що робить інтерпретацію менш інтуїтивною [31]. Під MSE розуміється віднімання прогнозованого значення від фактичного: похибка прогнозу = фактичне значення – прогнозоване значення

Загалом формула для розрахунку MSE є середнім значенням всіх абсолютних помилок:

$$MSE = \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}, \quad (3.2)$$

де $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ – прогнозовані значення; $y_1, y_2, \dots, y_n$ – справжні значення; n – кількість значень.

Описані вище методи є кількісним аналізом і будуть використовуватись для порівняльного аналізу рекомендаційних алгоритмів у наступних пунктах. Ці методи чудово підійдуть для колабораційної фільтрації і гібридної системи, адже в них були розраховані прогнозовані оцінки користувачів.

Для аналізування контент фільтрації буде використано якісний аналіз, в ході якого будуть розглянуті та порівняні отримані рекомендації від створених рекомендаційних алгоритмів.

3.2 Кількісний аналіз

Для розрахунку RMSE і MSE для колаборативної фільтрації та гібридної системи було використано відповідні методи з бібліотки sklearn. Ці показники визначались для вибірки з п'яти користувачів з персональними ідентифікаторами 10, 20, 30, 40, 50. Отримані значення для колаборативної фільтрації зображені на рисунку 3.1.

```
Collaborative filtering RMSE for first userID [10, 20, 30, 40, 50]: [1.1385113857674118, 1.1038584705897296, 1.035617502242628, 1.1212884944466344, 1.3135207383910572]
Collaborative filtering average RMSE: 1.1425593182874922
Collaborative filtering MSE for first userID [10, 20, 30, 40, 50]: [1.2962081755220327, 1.218503523092697, 1.0725036109512596, 1.2572878877784002, 1.725336730183388]
Collaborative filtering average MSE: 1.3139679855055555
```

Рис. 3.1. RMSE і MSE для колаборативної фільтрації

Отримані значення в контексті рекомендаційних систем є в межах норми. RMSE в середньому має значення 1.14, що свідчить про те, що прогнозовані оцінки зазвичай відрізняються на 1.14 бали.

RMSE і MSE для гібридної системи зображені на рисунку 3.2.

```
Hybrid RMSE for first userID [10, 20, 30, 40, 50]: [0.8659667208376395, 0.8882073983896642, 0.8624101883436535, 0.8251217358956104, 0.8627959473830049]
Hybrid average RMSE: 0.8609003981699145
Hybrid MSE for first userID [10, 20, 30, 40, 50]: [0.7498983615982943, 0.7889123825541356, 0.7437513329589359, 0.6808258790473856, 0.744416846820537]
Hybrid average MSE: 0.7415609605958576
```

Рис. 3.2. RMSE і MSE для гібридної системи

Показники для гібридної системи є значно кращими. Середнє значення RMSE по вибірці становить 0.86, що є дуже хорошим показником для рекомендаційної системи. Для кращого та більш наочного порівняння було створено діаграму, зображену на рисунку 3.3.

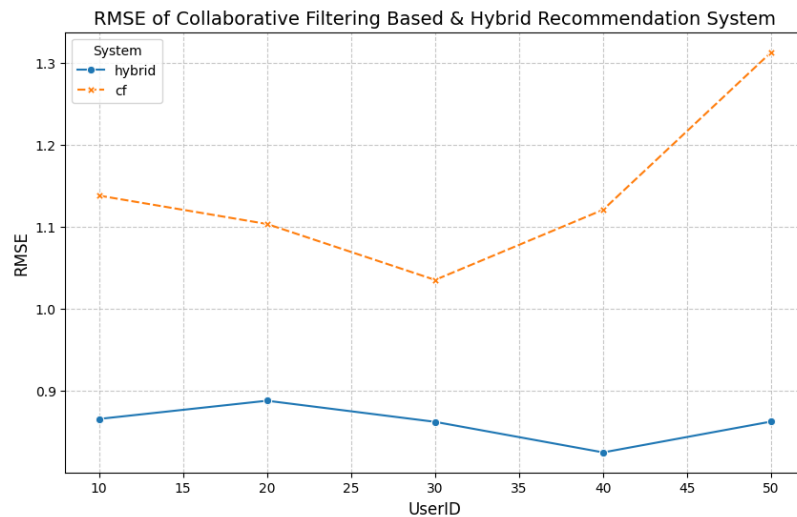


Рис. 3.3. RMSE для колаборативної фільтрації і гібридної системи

На зображеній вище діаграмі показані значення RMSE для кожного користувача з вибірки. По відношенню до колаборативної фільтрації гібридна система загалом має нижчий показник. Гістограма, зображена на рисунку 3.4, показує відношення середніх значень RMSE.

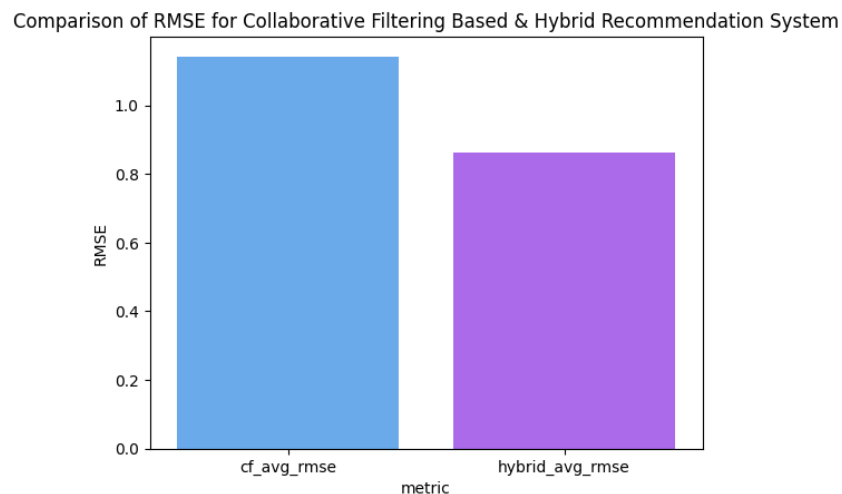


Рис. 3.4. Гістограма середніх значень RMSE вибірки

Показники MSE також були відображені у формі графіку на рисунку 3.5.

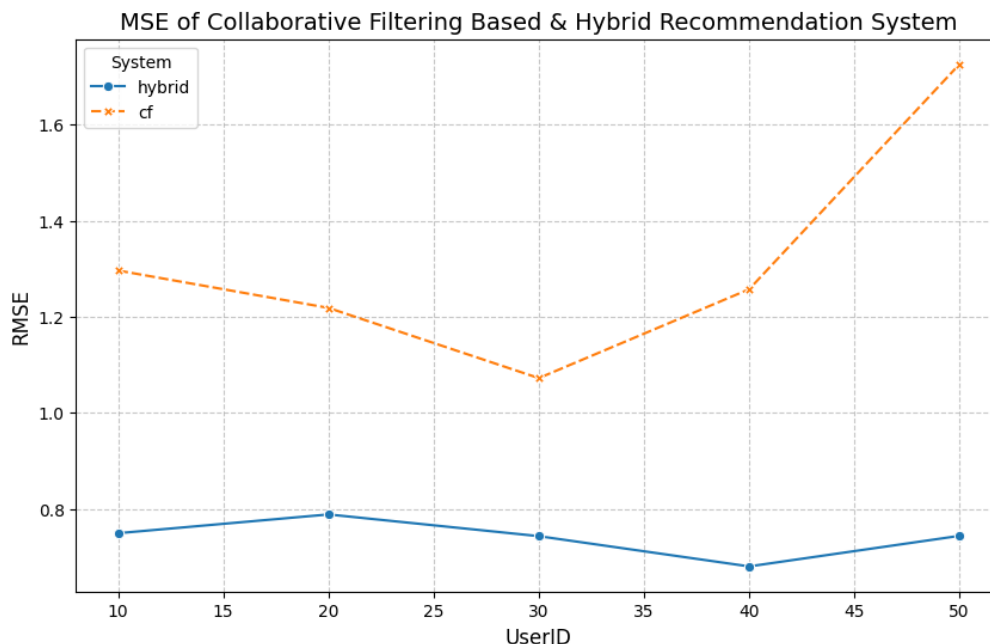


Рис. 3.5. MSE для колаборативної фільтрації і гібридної системи

З графіку знову ж такими видно набагато нижчий показник MSE для гібридної системи. Гістограма середнього значення MSE зображена на рисунку 3.6.

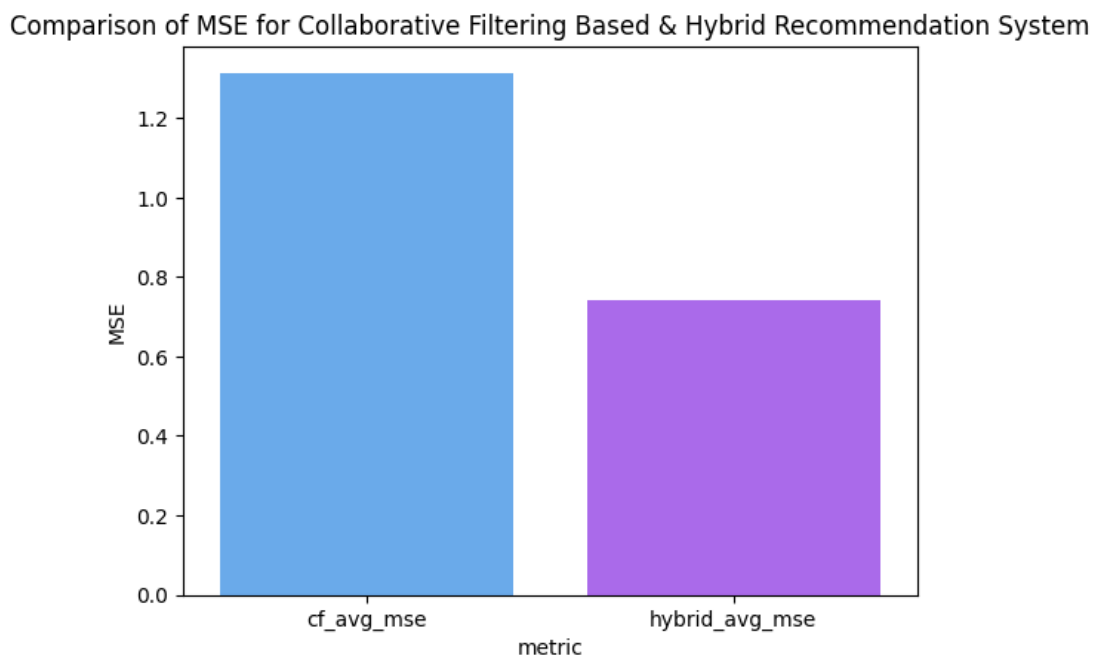


Рис. 3.6. Гістограма середнього значення MSE

Загалом, з проробленого аналізу та порівняння колаборативної фільтрації та гібридної системи явно видно, що краща якість рекомендацій надається гібридною системою.

3.3 Якісний аналіз

Рисунок 3.7 зображає рекомендації, створені з використанням колаборативної фільтрації для користувача з персональним ідентифікатором 20. Ці рекомендації базуються на попередніх вподобаннях користувача. Для їх створення було спрогнозовано рейтинги фільмів, які може поставити користувач.

	movieid	estimated_rating	title	actual_rating	genres	
	922	3083	4.899694	[Mr. Smith Goes to Washington]	[]	[Comedy, Drama]
	1674	142115	4.873062	[Five Dances]	[]	[Drama, Romance]
	1922	74727	4.818548	[Long Pigs]	[]	[Crime, Horror]
	4409	26596	4.805543	[Johnny Guitar]	[]	[Drama, Western]
	203	82	4.798701	[Miami Vice]	[]	[Action, Adventure, Crime, Thriller]
	54	296	4.797345	[Terminator 3: Rise of the Machines]	[4.5]	[Action, Thriller, Science Fiction]
	1096	314	4.784358	[Catwoman]	[]	[Action, Crime]
	122	4979	4.780184	[Windows on Monday]	[4.0]	[Drama]
	2203	52767	4.779155	[The Hunter]	[]	[Drama, Thriller]
	1612	665	4.767986	[Ben-Hur]	[]	[Action, Adventure, Drama, History]
	371	31658	4.767480	[Hour of the Gun]	[]	[Action, Drama, Western]
	66	5618	4.763355	[Cousin, Cousine]	[4.5]	[Romance, Comedy]
	64	318	4.760886	[The Million Dollar Hotel]	[]	[Drama, Thriller]
	455	265	4.755916	[Lili Marleen]	[]	[War, Drama, Romance]
	1667	668	4.755606	[On Her Majesty's Secret Service]	[]	[Adventure, Action, Thriller]
	426	52885	4.746248	[The Red and the White]	[5.0]	[Drama, War, Foreign]
	1037	98491	4.738320	[Wuthering Heights]	[]	[Drama, Romance]
	56	750	4.725053	[Murder She Said]	[]	[Drama, Crime, Mystery, Comedy]
	278	1914	4.724644	[FC Venus]	[]	[Comedy]
	2489	120478	4.721872	[The Haunting of Whaley House]	[]	[Horror]

Рис. 3.7. Рекомендації колаборативної фільтрації

Загалом рекомендовані фільми мають різноманітні жанри, що дає широкий вибір. Прогнозоване значення рейтингу, розраховане системою, до фільмів, які користувач вже оцінив є наближеним (різниця в середньому становить 0.2).

Рекомендації від системи на основі контент фільтрації для жанрів зображені на рисунку 3.8 і були утворені за фільмом «Harry Potter and the Prisoner of Azkaban».

	movie_index	similarity_score	title	movieid	genres
0	4742	1.000000	Harry Potter and the Philosopher's Stone	671	[[Adventure', 'Fantasy', 'Family']]
1	5381	1.000000	Clash of the Titans	10323	[[Adventure', 'Fantasy', 'Family']]
2	5653	1.000000	Harry Potter and the Chamber of Secrets	672	[[Adventure', 'Fantasy', 'Family']]
3	6073	1.000000	Journey to the Center of the Earth	11571	[[Adventure', 'Fantasy', 'Family']]
4	6982	1.000000	Peter Pan	10601	[[Adventure', 'Fantasy', 'Family']]
5	7696	1.000000	Harry Potter and the Prisoner of Azkaban	673	[[Adventure', 'Fantasy', 'Family']]
6	10519	1.000000	Harry Potter and the Goblet of Fire	674	[[Adventure', 'Fantasy', 'Family']]
7	10859	1.000000	Bröderna Lejonhjärta	11291	[[Adventure', 'Fantasy', 'Family']]
8	11942	1.000000	Jack the Giant Killer	25360	[[Adventure', 'Fantasy', 'Family']]
9	13841	1.000000	Harry Potter and the Half-Blood Prince	767	[[Adventure', 'Fantasy', 'Family']]
10	14775	1.000000	Percy Jackson & the Olympians: The Lightning T...	32657	[[Adventure', 'Fantasy', 'Family']]
11	16048	1.000000	Harry Potter and the Deathly Hallows: Part 1	12444	[[Adventure', 'Fantasy', 'Family']]
12	23252	1.000000	Neverland	106938	[[Adventure', 'Fantasy', 'Family']]
13	27433	1.000000	Cesta do pravěku	8779	[[Adventure', 'Fantasy', 'Family']]
14	37464	1.000000	Starik Khottabych	71103	[[Adventure', 'Fantasy', 'Family']]
15	32041	0.821258	Arthur 3: la guerre des deux mondes	42949	[[Animation', 'Adventure', 'Fantasy', 'Family']]
16	36688	0.821258	Pohádka o Honzíkovi a Mařence	51902	[[Animation', 'Adventure', 'Fantasy', 'Family']]
17	8530	0.798654	The Wonderful World of the Brothers Grimm	28367	[[Fantasy', 'Family']]
18	9631	0.798654	Alice in Wonderland	30923	[[Fantasy', 'Family']]
19	18745	0.798654	Don't Eat the Pictures: Sesame Street at the M...	319656	[[Fantasy', 'Family']]

Рис. 3.8. Рекомендації контент фільтрації за жанрами

З утвореної таблиці явно видно, що переважно рекомендуються фільми з тим самим набором жанрів, що може призвести до неточних рекмондацій, адже не завжди фільми схожі за жанрами мають схожий сюжет або ті унікальні риси, які подобаються користувачу.

Рекомендаційна система на основі контент фільтрації за описом надає надзвичайно унікальні рекомендації, проте показник схожості тільки наближений до одиниці. Список рекомендацій наведений на рисунку 3.9.

movie_index	similarity_score	title	movieid	genres	
0	18185	0.977138	Intruders	63207	[[Horror', 'Thriller', 'Drama']]
1	19499	0.975216	Astérix & Obélix - Au service de Sa Majesté	99770	[[Family', 'Adventure', 'Comedy']]
2	32266	0.973830	ストリートファイターZERO	17433	[[Action', 'Animation']]
3	13246	0.973410	The Dungeonmaster	27244	[[Horror', 'Fantasy', 'Thriller', 'Science Fi...
4	15737	0.973291	Infestation	22476	[[Action', 'Comedy', 'Horror', 'Science Ficti...
5	26057	0.971813	The Land Before Time VII: The Stone of Cold Fire	24102	[[Adventure', 'Fantasy', 'Animation', 'Scienc...
6	22571	0.971474	El caníbal	84362	[[Horror']]
7	5424	0.971089	Ernest Scared Stupid	32685	[[Horror', 'Comedy', 'Family']]
8	19995	0.970739	るろうに剣心	127533	[[Action', 'Drama', 'History']]
9	14861	0.970567	ルパン三世 sweet lost night ~魔法のランプは悪夢の予感~	59719	[[Action', 'Animation', 'Comedy', 'TV Movie']]
10	41813	0.970553	聖闘士星矢 真紅の少年伝説	66105	[[Action', 'Animation', 'Fantasy']]
11	26832	0.970549	Voodoo Possession	245589	[[Mystery', 'Horror', 'Drama']]
12	26080	0.970288	La Mission	43219	[[Drama']]
13	3852	0.969904	Vertical Limit	11678	[[Action', 'Adventure', 'Thriller']]
14	3188	0.969896	What Planet Are You From?	10416	[[Comedy', 'Drama', 'Romance', 'Science Ficti...
15	29063	0.969735	Белка и Стрелка: Лунные приключения	260310	[[Adventure', 'Animation', 'Comedy', 'Family']]
16	28378	0.969682	The Hollow	35946	[[Horror']]
17	36315	0.969503	Boris Sans Béatrice	372985	[[Drama']]
18	17715	0.969459	Warrior	59440	[[Action', 'Drama']]
19	23305	0.969316	Tangled Ever After	82881	[[Animation', 'Comedy', 'Action', 'Family']]

Рис. 3.9. Рекомендації контент фільтрації за описом

Наведений вище список був створений для фільму «Harry Potter and the Prisoner of Azkaban» і містить схожі за описом фільми, які не були присутні у контент фільтрації за описом.

Гібридна система, яка використовує контент фільтрацію за описом та колаборативну фільтрацію, надала якісні рекомендації для користувача з ідентифікатором 20 для фільму «Harry Potter and the Prisoner of Azkaban». Список її рекомендацій зображений на рисунку 3.10.

	movieid	similarity_score	title	estimated_rating	actual_rating
3	6073	1.000000	Journey to the Center of the Earth	4.358006	□
2	5653	1.000000	Harry Potter and the Chamber of Secrets	4.010744	□
0	4742	1.000000	Harry Potter and the Philosopher's Stone	3.872348	□
1	5381	1.000000	Clash of the Titans	3.872348	□
4	6982	1.000000	Peter Pan	3.872348	□
5	7696	1.000000	Harry Potter and the Prisoner of Azkaban	3.872348	□
6	10519	1.000000	Harry Potter and the Goblet of Fire	3.872348	□
7	10859	1.000000	Bröderna Lejonhjärta	3.872348	□
8	11942	1.000000	Jack the Giant Killer	3.872348	□
9	13841	1.000000	Harry Potter and the Half-Blood Prince	3.872348	□
10	14775	1.000000	Percy Jackson & the Olympians: The Lightning T...	3.872348	□
11	16048	1.000000	Harry Potter and the Deathly Hallows: Part 1	3.872348	□
12	23252	1.000000	Neverland	3.872348	□
13	27433	1.000000	Cesta do pravěku	3.872348	□
14	37464	1.000000	Starik Khottabych	3.872348	□
15	32041	0.821258	Arthur 3: la guerre des deux mondes	3.872348	□
16	36688	0.821258	Pohádka o Honzíkoví a Mařence	3.872348	□
17	8530	0.798654	The Wonderful World of the Brothers Grimm	3.872348	□
18	9631	0.798654	Alice in Wonderland	3.872348	□
19	18745	0.798654	Don't Eat the Pictures: Sesame Street at the M...	3.872348	□

Рис. 3.10. Рекомендації гібридної системи

Показник схожості для перших 15 фільмів становить 1, що є дуже хорошим показником. Загалом рекомендовані фільми є схожими до введених даних.

Після проведення кількісного та якісного аналізу, явну перевагу має гібридна система, яка надає рекомендації базуючись не лише на конкретному фільмі, а й на попередній історії вподобань користувача, що сприяє підвищенні якості рекомендацій.

3.4 Подальша робота

Рекомендаційні системи не можуть бути універсальними через складність і динамічність потреб користувачів, які постійно змінюються під впливом

зовнішніх факторів, нових уподобань або змін у поведінці. Кожен користувач має унікальні інтереси, контекст споживання інформації та очікування від системи. Окрім того, смаки користувачів можуть еволюціонувати з часом, і те, що було актуальним раніше, може втратити свою цінність у майбутньому. У зв'язку з цим, рекомендаційні системи мають бути адаптивними, щоб враховувати не лише поточні переваги, але й тенденції, які формуються у поведінці користувачів.

Для покращення роботи рекомендаційних систем можна інтегрувати додаткові елементи персоналізації на основі контексту. Наприклад, врахування геолокації, часу доби або дня тижня дозволяє надавати більш релевантні рекомендації залежно від обставин користувача. Також важливим є використання неявних відгуків, таких як наприклад час проведений на сторінці предмету, або перегляду відео.

Саме впровадження гібридної системи надасть найкращий результат для користувачів і дозволить розробку рекомендаційної системи, яка зможе надавати актуальні пропозиції різним користувачам.

Адаптивність та інновації в архітектурі рекомендаційних систем мають стати пріоритетом для їхнього розвитку. Оскільки потреби користувачів є складними та багатогранними, необхідно постійно вдосконалювати алгоритми, враховувати нові джерела даних і забезпечувати інтерактивність. Тільки так можна гарантувати, що рекомендаційна система залишатиметься ефективною і актуальною у сучасному цифровому світі.

Висновки до третього розділу

У цьому розділі було проведено детальний аналіз розроблених алгоритмів рекомендаційних систем. Було визначено та описано критерії їх оцінювання.

У кількісному аналізі було порівняно ефективність роботи колаборативної фільтрації та гібридної системи з використанням таких метрик, як RMSE та MSE.

У якісному аналізі було ретельно розглянуто результати створених рекомендацій усіма розроблених алгоритмів. За результатами аналізу найкращі показники показав гібридний підхід рекомендаційної системи.

Було також наведено можливі способи покращення рекомендаційних систем.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було проведено дослідження рекомендаційних систем, принципів їх роботи, детально описано методи та алгоритми для створення рекомендаційних систем, успішно реалізовано розглянуті алгоритми та проведено їх аналіз.

В ході виконання дослідження було проведено загальний огляд рекомендаційних систем, описано принципи їхньої роботи, сфери застосування, основні види та ключові проблеми. Детально розглянуто кожен тип рекомендаційних систем. Для колаборативної фільтрації описано загальний принцип функціонування, проаналізовано особливості підтипів, зокрема було описано підтипи на основі пам'яті та на основі моделі. Також пояснено метод матричної факторизації, який використовується у колаборативній фільтрації на основі моделі. Було розглянуто схему роботи контент фільтрації, наведено принципи роботи методів TF-IDF та Word2Vec, які використовуються для векторизації тексту, а також детально розглянуто процес розрахунку оцінки схожості. Окрім цього, представлено й проаналізовано шість підходів для створення гібридних рекомендаційних систем.

Далі було обрано набір даних з інформацією про фільми від MovieLens, проведено його обробку та розвідувальний аналіз. Було реалізовано чотири типи рекомендаційних систем. Зокрема, впроваджено колаборативну фільтрацію із застосуванням методу SVD, детально описано принцип роботи написаного алгоритму та представлено результати його застосування у формі таблиці рекомендацій, яка містить інформацією про фільми. Для контент фільтрації було реалізовано дві системи: перша на основі аналізу жанрів із використанням методу TF-IDF, друга — на основі аналізу описів, для чого було поєднано TF-IDF із Word2Vec. У роботі наведено принципи функціонування обох підходів та проаналізовано результати їх застосування. Гібридну рекомендаційну систему було розроблено з використанням каскадного підходу. На першому етапі

застосовувалася контентна фільтрація за жанрами, після чого на отриманих результатах проводилася колаборативна фільтрація.

Також в ході виконання роботи було проведено детальний аналіз розроблених алгоритмів рекомендаційних систем. Було визначено та описано критерії для їх оцінювання. У кількісному аналізі було порівняно ефективність роботи колаборативної фільтрації та гібридної системи з використанням таких метрик, як RMSE та MSE. У якісному аналізі було ретельно розглянуто та описано результати створення рекомендацій усіма розроблених алгоритмів. За результатами аналізу найкращі показники має гібридний підхід рекомендаційної системи. Також наведено можливі способи покращення рекомендаційних систем.

Отже, завдання на кваліфікаційну роботу виконано. Були успішно розроблені та проаналізовані чотири типи рекомендаційних систем і визначено гібридний підхід як найкращий.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sachi Nandan Mohanty. Recommender System with Machine Learning and Artificial Intelligence / Sachi Nandan Mohanty, Jyotir Moy Chatterjee, Sarika Jain, Ahmed A. Elngar, Priya Gupta – Wiley-Scrivener, 2020.
2. The Amazon Recommendations Secret to Selling More Online [Електронний ресурс] – Режим доступу до ресурсу: <https://www.rejoiner.com/resources/amazon-recommendations-secret-selling-online>
3. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. – J Big Data 9, 2022 – с. 59
4. Recommendation System [Електронний ресурс] – Режим доступу до ресурсу: <https://www.nvidia.com/en-us/glossary/recommendation-system/>
5. Yehuda Koren. Advances in Collaborative Filtering / Yehuda Koren, Steffen Rendle, Robert Bell. – Recommender Systems Handbook, 2022
6. Deep Dive on KNN: Understanding and Implementing the K-Nearest Neighbors Algorithm [Електронний ресурс] – Режим доступу до ресурсу: <https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/>
7. Ricci. Recommender Systems Handbook. / Springer. – 2022.
8. Matrix Factorization Techniques For Recommender Systems [Електронний ресурс] – Режим доступу до ресурсу: [https://datajobs.com/data-science-repo/Recommender-Systems-\[Netflix\].pdf](https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf)
9. Kim Falk. Practical Recommender Systems. / Kim Falk – Manning Publications, 2019.
10. Netflix Update: Try This at Home [Електронний ресурс] – Режим доступу до ресурсу: <https://sifter.org/~simon/journal/20061211.html>
11. Python 3.13.1 documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/>
12. What is collaborative filtering? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/think/topics/collaborative-filtering>

13. What is content-based filtering? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.ibm.com/think/topics/content-based-filtering>
14. Content Based Recommendation: How It Works and Why You Should Use It? [Электронный ресурс] – Режим доступа до ресурсу: https://redfield.ai/content-based-recommendation/#Method_1_The_Vector_Space_Method
15. Speech and Language Processing [Электронный ресурс] – Режим доступа до ресурсу: <https://web.stanford.edu/~jurafsky/slp3/14.pdf>
16. Turan, B. Embeddings with Word2Vec in non-NLP Contexts – Towards Data Science, 2019.
17. The Illustrated Word2Vec [Электронный ресурс] – Режим доступа до ресурсу: <http://jalammar.github.io/illustrated-word2vec/>
18. Continuous bag of words (CBOW) in NLP [Электронный ресурс] – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/continuous-bag-of-words-cbow-in-nlp/>
19. Word Embeddings: CBOW vs Skip-Gram [Электронный ресурс] – Режим доступа до ресурсу: <https://www.baeldung.com/cs/word-embeddings-cbow-vs-skip-gram>
20. Hybrid Recommender Systems: Beginner's Guide [Электронный ресурс] – Режим доступа до ресурсу: <https://marketsy.ai/blog/hybrid-recommender-systems-beginners-guide>
21. The Movies Dataset [Электронный ресурс] – Режим доступа до ресурсу: <https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset>
22. pandas documentation [Электронный ресурс] – Режим доступа до ресурсу: <https://pandas.pydata.org/docs/>
23. Matplotlib: Visualization with Python [Электронный ресурс] – Режим доступа до ресурсу: <https://matplotlib.org/>.
24. seaborn: statistical data visualization [Электронный ресурс] – Режим доступа до ресурсу: <https://seaborn.pydata.org/>
25. Matrix Factorization-based algorithms [Электронный ресурс] – Режим доступа до ресурсу: https://surprise.readthedocs.io/en/stable/matrix_factorization.html

26. scikit-learn [Электронный ресурс] – Режим доступа до ресурсу: <https://scikit-learn.org/stable/>
27. Building a Recommendation System using Word2vec [Электронный ресурс] – Режим доступа до ресурсу: <https://www.analyticsvidhya.com/blog/2019/07/how-to-build-recommendation-system-word2vec-python/>
28. Gensim Word2Vec embeddings [Электронный ресурс] – Режим доступа до ресурсу: <https://radimrehurek.com/gensim/models/word2vec.html>
29. 10 metrics to evaluate recommender and ranking systems [Электронный ресурс] – Режим доступа до ресурсу: <https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems>
30. Root Mean Squared Error (RMSE) [Электронный ресурс] – Режим доступа до ресурсу: https://help.sap.com/docs/SAP_PREDICTIVE_ANALYTICS/41d1a6d4e7574e32b815f1cc87c00f42/5e5198fd4afe4ae5b48fefe0d3161810.html
31. Mean Squared Error (MSE) [Электронный ресурс] – Режим доступа до ресурсу: <https://statisticsbyjim.com/regression/mean-squared-error-mse/>

ДОДАТКИ

Розвідувальний аналіз movies.csv

```
import pandas as pd

import seaborn as sns

import numpy as np

import warnings

import matplotlib.pyplot as plt

import os

import nltk

# Movies: data clean up

movies_initial = pd.read_csv('movies-metadata/movies_metadata.csv')

display(movies_initial.shape)

display(movies_initial.columns)

display(movies_initial.head())

movies_initial.dtypes

movies = movies_initial.drop(['budget', 'adult', 'belongs_to_collection', 'homepage',
                             'spoken_languages', 'tagline', 'video', 'poster_path', 'status',
                             'vote_count', 'popularity', 'imdb_id', 'revenue'], axis=1)

movies.sample(5)

movies.isnull().sum()

movies = movies.dropna()

movies.isnull().sum()

display(movies[movies['runtime']==0.0].count()['runtime'])

movies.shape

movies['genres'] = movies['genres'].apply(lambda x: ', '.join([d['name'] for d in eval(x)]) if x else '')

movies['production_companies'] = movies['production_companies'].astype(str)

movies['production_companies'] = movies['production_companies'].apply(lambda x: ', '.join([d['name'] for d in
eval(x)]) if x else '')

movies['production_countries'] = movies['production_countries'].apply(lambda x: ', '.join([d['name'] for d in
eval(x)]) if x else '')
```

```

movies['release_date'] = movies['release_date'].apply(lambda x: x[:4])

movies.rename(columns={'id': 'movieId', 'release_date': 'release_year'}, inplace=True)

movies = movies.astype({'movieId': 'int64'})

movies.to_csv("movies-metadata/movies.csv", index=False)

# Average Ratings

plt.figure(figsize=(10,6))

sns.histplot(movies['vote_average'], bins=30, kde=True, color='skyblue')

plt.title('Distribution of Average Ratings')

plt.xlabel('Average Ratings')

plt.ylabel('Number of Films')

plt.show()


# Top genres

df_exploded = movies['genres'].str.split(' ', expand=True).stack().reset_index(level=1, drop=True)

df_exploded.name = 'genre'

genre_counts = df_exploded.value_counts()

plt.figure(figsize=(10, 6))

sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='cool')

plt.title('Top Genres by Frequency', fontsize=16)

plt.xlabel('Frequency', fontsize=12)

plt.ylabel('Genre', fontsize=12)

plt.show()


# Top 5 most common languages

top_languages = movies['original_language'].value_counts().head(5)

plt.figure(figsize=(8, 8))

plt.pie(top_languages, labels=top_languages.index, autopct='%1.1f%%', startangle=140,
        colors=plt.cm.Paired.colors)

plt.title('Most Common Languages in Movies', fontsize=16)

plt.show()


# Average rating for top 5 genres

```

```
genre_ratings = movies.assign(genres=movies['genres'].str.split(', ')).explode('genres')

genre_counts = genre_ratings['genres'].value_counts()

top_5_genres = genre_counts.head(10).index

top_genre_ratings = genre_ratings[genre_ratings['genres'].isin(top_5_genres)]

avg_ratings = top_genre_ratings.groupby('genres')['vote_average'].mean().reset_index()

sns.barplot(data=avg_ratings, x='genres', y='vote_average', palette='cool')

plt.title('Average Rating for Top 5 Most Common Genres')

plt.ylabel('Average Rating')

plt.xlabel('Genres')

plt.xticks(rotation=45)

plt.show()
```

Колаборативна фільтрація

```
from surprise import Reader, Dataset, SVD

from surprise.model_selection import cross_validate

reader = Reader()

data = Dataset.load_from_df(filtered_ratings[['userId', 'movieId', 'rating']], reader)

svd = SVD()

x = cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

titles = movies['original_title']


def cf_top_rec(userId, ratings):

    cf_df = pd.DataFrame()

    movie_id_list = ratings['movieId'].unique().tolist()

    est = []

    movie_name_list = []

    actual_score_list = []

    genre_list = []

    for i in movie_id_list:

        pred = svd.predict(userId, i, verbose=False)

        est.append(pred[3])

        movie_name = movies[movies.movieId == i]['title'].values

        movie_name_list.append(movie_name)

        actual_score = ratings[(ratings.movieId==i)&(ratings.userId==userId)]['rating'].values

        actual_score_list.append(actual_score)

        genre = movies[movies.movieId == i]['genres'].values

        try:

            genre_list.append(genre)

        except:

            genre_list.append('Unavailable')
```

```
cf_df['movieId'] = movie_id_list

cf_df['estimated_rating'] = est

cf_df['title'] = movie_name_list

cf_df['actual_rating'] = actual_score_list

cf_df['genres'] = genre_list

cf_df = cf_df.sort_values(['estimated_rating'], ascending=False)

return cf_df


userId = 20

cf_top_rec(userId, filtered_ratings).head(20)
```

Контент фільтрація для жанрів

```

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear_kernel

from sklearn.metrics.pairwise import cosine_similarity

movies['genres'] = movies['genres'].str.split(', ')

movies['genres'] = movies['genres'].fillna("").astype('str')

tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), stop_words='english')

tfidf_matrix = tf.fit_transform(movies['genres'])

df = pd.DataFrame(tfidf_matrix.toarray(), columns=tf.get_feature_names_out())

display(df)

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

cosine_sim[:5, :5]

titles = movies['original_title']

movie_id = movies['movieId']

indices = pd.Series(movies.index, index=movies['title'])

def genre_rec(title):

    idx = indices[title]

    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:]

    movie_indices = [i[0] for i in sim_scores]

    cb_df = pd.DataFrame(sim_scores)

    cb_df['title'] = titles.iloc[movie_indices].values

    cb_df['movieId'] = movie_id.iloc[movie_indices].values

    cb_df = cb_df.rename(columns={0: "movie_index", 1: "similarity_score"})

    genre_list = []

    for i in cb_df['movieId'].values:

        genre = movies[movies.movieId==i]['genres'].values

```

```
try:

    genre_list.append(genre)

except:

    genre_list.append('Unavailable')

cb_df['genres'] = genre_list

return cb_df


title = 'Harry Potter and the Prisoner of Azkaban'

dfrec = genre_rec(title)


display(dfrec.head(20))
```

Контент фільтрація для описів

```
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import RegexpTokenizer

import re

from gensim.models import Word2Vec

from gensim.models.phrases import Phrases, Phraser

from gensim.models import KeyedVectors

# Overview cleanup


def _removeNonAscii(s):

    return "".join(i for i in s if ord(i)<128)


def make_lower_case(text):

    return text.lower()


def remove_stop_words(text):

    text = text.split()

    stops = set(stopwords.words("english"))

    text = [w for w in text if not w in stops]

    text = " ".join(text)

    print(text)

    return text


def remove_html(text):

    html_pattern = re.compile('<.*?>')

    return html_pattern.sub(r'', text)
```

```

def remove_punctuation(text):

    tokenizer = RegexpTokenizer(r'\w+')

    text = tokenizer.tokenize(text)

    text = " ".join(text)

    return text

movies['overview'] = movies['overview'].astype(str)

movies['cleaned'] = movies['overview'].apply(_removeNonAscii)

movies['cleaned'] = movies['cleaned'].apply(func = make_lower_case)

movies['cleaned'] = movies['cleaned'].apply(func = remove_stop_words)

movies['cleaned'] = movies['cleaned'].apply(func = remove_punctuation)

movies['cleaned'] = movies['cleaned'].apply(func = remove_html)

display(movies.head())

# Bigram distribution

tf = TfidfVectorizer(ngram_range=(2, 2), stop_words='english', lowercase = False)

tfidf_matrix = tf.fit_transform(movies['cleaned'])

total_words = tfidf_matrix.sum(axis=0)

freq = [(word, total_words[0, idx]) for word, idx in tf.vocabulary_.items()]

freq = sorted(freq, key = lambda x: x[1], reverse=True)

bigram = pd.DataFrame(freq)

bigram.rename(columns = {0: 'bigram', 1: 'count'}, inplace = True)

bigram = bigram.head(20)

bigram.plot(x='bigram', y='count', kind = 'bar', title = "Bigram distribution for the top 20 words in the
movie description", figsize = (15,7), )

# Trigram distribution

tf = TfidfVectorizer(ngram_range=(3, 3), stop_words='english', lowercase = False)

tfidf_matrix = tf.fit_transform(movies['overview'])

total_words = tfidf_matrix.sum(axis=0)

```

```

freq = [(word, total_words[0, idx]) for word, idx in tf.vocabulary_.items()]

freq =sorted(freq, key = lambda x: x[1], reverse=True)

trigram = pd.DataFrame(freq)

trigram.rename(columns = {0:'trigram', 1: 'count'}, inplace = True)


trigram = trigram.head(20)

trigram.plot(x='trigram', y='count', kind = 'bar', title = "Trigram distribution for the top 20 words in the
movie description", figsize = (15,7), )

EMBEDDING_FILE = 'GoogleModel/GoogleNews-vectors-negative300.bin.gz'

i = 0

corpus = []

for words in movies['cleaned']:

    corpus.append(words.split())

# Training corpus with Google Pretrained Model

google_model = Word2Vec(vector_size = 300, window=5, min_count = 2, workers = 2)

google_model.build_vocab(corpus)

google_model.wv.load_word2vec_format(EMBEDDING_FILE, binary=True)

google_model.train(corpus, total_examples=google_model.corpus_count, epochs = 5)

#Building TFIDF model and calculate TFIDF score

tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df = 5, stop_words='english')

tfidf.fit(movies['cleaned'])

tfidf_list = dict(zip(tfidf.get_feature_names_out(), list(tfidf.idf_)))

tfidf_feature = tfidf.get_feature_names_out()


# Building TF-IDF Word2Vec

tfidf_vectors = []

line = 0

for desc in corpus:

    sent_vec = np.zeros(300)

    weight_sum =0

    for word in desc:

```

```

if word in google_model.wv.key_to_index and word in tfidf_feature:

    vec = google_model.wv[word]

    tf_idf = tfidf_list[word] * (desc.count(word) / len(desc))

    sent_vec += (vec * tf_idf)

    weight_sum += tf_idf

if weight_sum != 0:

    sent_vec /= weight_sum

tfidf_vectors.append(sent_vec)

line += 1

cosine_similarities = cosine_similarity(tfidf_vectors, tfidf_vectors)

def desc_rec(title):

    movies_df = movies[['title', 'overview']]

    indices = pd.Series(movies.index, index = movies['title']).drop_duplicates()

    idx = indices[title]

    sim_scores = list(enumerate(cosine_similarities[idx]))

    sim_scores = sorted(sim_scores, key = lambda x: x[1], reverse = True)

    sim_scores = sim_scores[1:]

    movie_indices = [i[0] for i in sim_scores]

    cb_desc_df = pd.DataFrame(sim_scores)

    cb_desc_df['title'] = titles.iloc[movie_indices].values

    cb_desc_df['movieId'] = movie_id.iloc[movie_indices].values

    cb_desc_df = cb_desc_df.rename(columns={0: "movie_index", 1: "similarity_score"})

    genre_list_desc = []

    for i in cb_desc_df['movieId'].values:

        genre = movies[movies.movieId==i]['genres'].values

        try:

            genre_list_desc.append(genre)

        except:

```

```
genre_list_desc.append('Unavailable')

cb_desc_df['genres'] = genre_list_desc

return cb_desc_df
```

Гібридна система

```

titles_hyb = movies['title']

indices = pd.Series(movies.index, index=movies['title'])

def hybrid_recs(userId, title, ratings):

    idx = indices[title]

    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:100]

    movie_indices = [i[0] for i in sim_scores]

    hybrid_df = pd.DataFrame(sim_scores)

    hybrid_df['title'] = titles.iloc[movie_indices].values

    est = []

    actual_score_list = []

    hybrid_df = hybrid_df.rename(columns={0: "movieId", 1: "similarity_score"})

    for i in movie_indices:

        pred = svd.predict(userId, i, verbose=False)

        est.append(pred[3])

        actual_score = ratings[(ratings.movieId==i)&(ratings.userId==userId)]['rating'].values

        actual_score_list.append(actual_score)

    hybrid_df['estimated_rating'] = est

    hybrid_df['actual_rating'] = actual_score_list

    hybrid_df = hybrid_df.sort_values(['similarity_score', 'estimated_rating'], ascending=[False,False])

    return hybrid_df

userId = 20

title = 'Harry Potter and the Prisoner of Azkaban'

hybrid_recs(userId, title, filtered_ratings).head(20)

```

Порівняння між колаборативною системою та гібридною

```
from sklearn.metrics import root_mean_squared_error, mean_squared_error

def calc_rmse(predictions, actual):

    return root_mean_squared_error(actual, predictions)

def calc_mse(predictions, actual):

    return mean_squared_error(actual, predictions)

cf_rmse_val = []

cf_mse_val = []

users_test = [10, 20, 30, 40, 50]

for i in users_test:

    x = cf_top_rec(i, train_ratings)

    est = []

    act = []

    est_df = pd.DataFrame

    est_df = x[['movieId', 'estimated_rating']].copy()

    actual_ratings = est_df.merge(test_ratings[['movieId', 'rating']], on='movieId', how='left')

    est_df['actual_rating'] = actual_ratings['rating'].copy()

    est_df = est_df.dropna()

    est = est_df['estimated_rating'].tolist()

    act = est_df['actual_rating'].tolist()

    cf_rmse_val.append(calc_rmse(est, act))

    cf_mse_val.append(calc_mse(est, act))

cf_avg_rmse = sum(cf_rmse_val)/len(cf_rmse_val)

cf_avg_mse = sum(cf_mse_val)/len(cf_mse_val)
```

```

print(f"Collaborative filtering RMSE for first userID [10, 20, 30, 40, 50]: {cf_rmse_val}")

print(f"Collaborative filtering average RMSE: {cf_avg_rmse}")

print(f"Collaborative filtering MSE for first userID [10, 20, 30, 40, 50]: {cf_mse_val}")

print(f"Collaborative filtering average MSE: {cf_avg_mse}")


hyb_rmse_val = []

hyb_mse_val = []


users_test = [10, 20, 30, 40, 50]


for i in users_test:

    x = hybrid_recs(i, 'Harry Potter and the Prisoner of Azkaban', train_ratings)

    est = []

    act = []

    est_df = x[['movieId', 'estimated_rating']].copy()

    actual_ratings = est_df.merge(test_ratings[['movieId', 'rating']], on='movieId', how='left')

    est_df['actual_rating'] = actual_ratings['rating'].copy()

    est_df = est_df.dropna()

    est = est_df['estimated_rating'].tolist()

    act = est_df['actual_rating'].tolist()

    hyb_rmse_val.append(calc_rmse(est, act))

    hyb_mse_val.append(calc_mse(est, act))


hyb_avg_rmse = sum(hyb_rmse_val)/len(hyb_rmse_val)

hyb_avg_mse = sum(hyb_mse_val)/len(hyb_mse_val)

```

```

print(f"Hybrid RMSE for userID [10, 20, 30, 40, 50]: {hyb_rmse_val}")

print(f"Hybrid average RMSE: {hyb_avg_rmse}")

print(f"Hybrid MSE for userID [10, 20, 30, 40, 50]: {hyb_mse_val}")

print(f"Hybrid average MSE: {hyb_avg_mse}")

# RMSE plot

rmse_df = pd.DataFrame({'userId' : users_test, 'hybrid' : hyb_rmse_val, 'cf': cf_rmse_val})

df_long = rmse_df.melt(id_vars='userId', var_name='System', value_name='RMSE')

plt.figure(figsize=(10, 6))

sns.lineplot(data=df_long, x='userId', y='RMSE', hue='System', style='System', markers=True, dashes=True)

plt.title('RMSE of Collaborative Filtering Based & Hybrid Recommendation System', fontsize=14)

plt.xlabel('UserID', fontsize=12)

plt.ylabel('RMSE', fontsize=12)

plt.grid(True, linestyle='--', alpha=0.7)

plt.show()

# Average RMSE histogramm

rmse_df_hist = pd.DataFrame({'metric': ['cf_avg_rmse', 'hybrid_avg_rmse'], 'rmse': [cf_avg_rmse,
hyb_avg_rmse]})

sns.barplot(x='metric', y='rmse', data=rmse_df_hist, palette='cool')

plt.ylabel('RMSE')

plt.title('Comparison of RMSE for Collaborative Filtering Based & Hybrid Recommendation System')

plt.show()

# MSE plot

mse_df = pd.DataFrame({'userId' : users_test, 'hybrid' : hyb_mse_val, 'cf': cf_mse_val})

df_long = mse_df.melt(id_vars='userId', var_name='System', value_name='MSE')

plt.figure(figsize=(10, 6))

sns.lineplot(data=df_long, x='userId', y='MSE', hue='System', style='System', markers=True, dashes=True)

plt.title('MSE of Collaborative Filtering Based & Hybrid Recommendation System', fontsize=14)

plt.xlabel('UserID', fontsize=12)

plt.ylabel('RMSE', fontsize=12)

plt.grid(True, linestyle='--', alpha=0.7)

```

```
plt.show()

# Average MSE histogramm

mse_df_hist = pd.DataFrame({'metric': ['cf_avg_mse', 'hybrid_avg_mse'], 'mse': [cf_avg_mse, hyb_avg_mse]})

sns.barplot(x='metric', y='mse', data=mse_df_hist, palette='cool')

plt.ylabel('MSE')

plt.title('Comparison of MSE for Collaborative Filtering Based & Hybrid Recommendation System')

plt.show()
```