Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.89/.048

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Розробка рекомендаційної системи на основі вмісту та спільної фільтрації ”

(*назва згідно з наказом ректора*)

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - 38.00.00.000

(*позначення*)

Студент

ІПЗ-43_____/ Поліна ЛАЗУКІНА/

(шифр групи) (підпис) (дата)(розшифровка підпису)

Науковий керівник

к.т.н., асист._____/Максим ТКАЧЕНКО/

(посада) (підпис) (дата) (розшифровка підпису)

Консультант
з питань нормоконтролю

фахівець_____/Тамара ЧАПОВСЬКА/

(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

д.т.н., проф._____/Олексій БИЧКОВ/

(посада) (підпис) (дата) (розшифровка підпису)

Київ – 2021

Рішенням Екзаменаційної комісії
випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

професор, доктор техн. наук Віктор ВИШНІВСЬКИЙ

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:
Завідувач кафедри програмних систем і технологій
_____(Олексій БИЧКОВ)
(підпис) (розшифровка підпису)
„___” _____ 2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
СТУДЕНТУ

Лазукіній Поліні Євгенівні
(прізвище, ім’я, по-батькові)

1. **Тема випускної кваліфікаційної бакалаврської роботи** “Розробка рекомендаційної системи на основі вмісту та спільної фільтрації”
   керівник проєкту (роботи) Ткаченко Максим Васильович, к.т.н., асистент
   затверджена наказом вищого навчального закладу від “ ” 20 р. №
2. **Строк подання студентом роботи** 2021 р.
3. **Вихідні дані до роботи** Науково-технічні публікації, присвячені розгляду машинного навчання, результати експериментальних досліджень, моделі побудови рекомендаційних систем. Форма діалогу: веб-сервіс. Перелік використовуваних програмних засобів: ОС Windows 10, інтегроване середовище розробки Visual Studio Code; CSS/HTML, Python, Flask framework.
4. **Зміст пояснювальної записки (перелік питань, що їх належить розробити)**
   1. Аналіз предметної галузі.
   2. Аналіз застосування рекомендаційних систем.
   3. Математичний опис створення контентної та колаборативної фільтрації.
   4. Проєктування рекомендаційної системи.
   5. Програмна реалізація проєкту.

6. Аналіз та порівняння результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Схема роботи традиційної системи рекомендацій (рис. 1.1, ст. 16)

2. Матриці взаємодій користувач-елемент (рис. 1.2, ст. 18)

3. Схема взаємодії користувача й елементів в СВ системі (рис. 1.3, ст. 21)

4. Діаграма використання системи рекомендацій (рис. 3.1, ст. 33)

5. Загальна клієнт-серверна схема (рис. 3.2, ст. 34)

6. Діаграма взаємодії користувача із системою рекомендацій (рис. 3.3, ст. 34)

7. Діаграма компонентів веб-застосунку рекомендацій (рис. 3.4, ст. 35)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основна частина	к.т.н., асист., Максим ТКАЧЕНКО		

7. Дата видачі завдання _____

Керівник _____ Максим ТКАЧЕНКО

(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Поліна ЛАЗУКІНА

(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1.	Отримання завдання на випускну кваліфікаційну роботу.	20.10.2020	виконано
2.	Аналіз завдання, пошук і вивчення відповідної літератури.	09.11.2020-30.11.2020	виконано
3.	Формування постановки задачі.	16.12.2020	виконано
4.	Дослідження й аналіз наявних стратегій створення рекомендаційних систем.	21.12.2020-04.01.2021	виконано

5.	Математичний опис методів.	13.01.2021-24.01.2021	виконано
6.	Проєктування рекомендаційного сервісу.	01.02.2021-20.02.2021	виконано
7.	Розробка алгоритму системи для надання рекомендацій.	22.02.2021-04.03.2021	виконано
8.	Розробка веб-засобу з використанням розробленого алгоритму надання рекомендацій.	08.03.2021-29.03.2021	виконано
9.	Аналіз та порівняння результатів, отриманих за допомогою програмного засобу.	29.03.2021-05.04.2021	виконано
10.	Оформлення пояснювальної записки та текстових матеріалів.	12.04.2021-22.05.2021	виконано
11.	Оформлення презентаційних матеріалів.	24.05.2021-30.05.2021	виконано

Студент – бакалавр _____ Поліна ЛАЗУКІНА

(підпис) (розшифровка підпису)

Керівник роботи _____ Максим ТКАЧЕНКО

(підпис) (розшифровка підпису)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 79 с., 18 рис., 2 табл., 5 додат., 13 джерел.

Тема: Розробка рекомендаційної системи на основі вмісту та спільної фільтрації.

Об'єкт дослідження: інтернет-сервіси, що включають процес генерації рекомендацій.

Мета роботи: виявлення нових закономірностей та методичне обґрунтування процесу створення системи рекомендацій у вигляді веб-застосунку, отримання найбільш релевантних рекомендацій, а також оцінка та порівняння результатів роботи; огляд наявних підходів до реалізації рекомендаційних систем, на підставі вивченого теоретичного матеріалу, розробити алгоритм формування content-based та collaborative рекомендаційної системи, використовуючи машинне навчання; на основі виявлених проблем запропонувати поліпшення методів.

Предмет дослідження: методи та моделі створення, користування особистих рекомендацій (контентна та колаборативна фільтрація).

Результати дослідження: описані можливості, застосування та оцінка методів, алгоритмів рекомендаційних систем. На підставі аналізу результатів системи, зроблений логічний підсумок спостережень та запропоновано поліпшення алгоритмів.

Висновок: у результаті досліджень отримано реалізовану стратегію надання рекомендацій, зручний веб-застосунок із вбудованою стратегією, яка може бути використана в будь-якому сучасному веб-сервісі, якщо ті потребують рекомендаційного механізму із застосування підходу вмісту та колаборативної фільтрації.

РЕКОМЕНДАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, КОНТЕНТНА ФІЛЬТРАЦІЯ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, PYTHON, ВЕБ-ЗАСТОСУНОК.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 79 с., 18 рис., 2 табл., 5 прил., 13 источников.

Тема: Разработка рекомендательной системы на основе содержания и совместной фильтрации.

Объект исследования: интернет-сервисы, включающие процесс генерации рекомендаций.

Цель работы: выявление новых закономерностей и методическое обоснование процесса создания системы рекомендаций в виде веб-приложения, получения наиболее релевантных рекомендаций, а также оценка и сравнение результатов работы; обзор существующих подходов к реализации рекомендательных систем, на основании изученного теоретического материала, разработать алгоритм формирования content-based и collaborative рекомендательной системы, используя машинное обучение; на основе выявленных проблем предложить улучшения методов.

Предмет исследования: методы и модели создания, использования личных рекомендаций (контентная и совместная фильтрация).

Результаты исследования: описаны возможности, применение и оценка методов, алгоритмов рекомендательных систем. На основании анализа результатов системы, сделан логический итог наблюдений и предложены улучшения алгоритмов.

Вывод: в результате исследований получено реализованную стратегию предоставления рекомендаций, удобное веб-приложение со встроенной стратегией, которая может быть использована в любом современном веб-сервисе, если те нуждаются в рекомендательном механизме по применению подхода содержания и коллаборативной фильтрации.

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, МАШИННОЕ ОБУЧЕНИЕ, КОНТЕНТНАЯ ФИЛЬТРАЦИЯ, КОЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ, PYTHON, ВЕБ-ПРИЛОЖЕНИЕ.

ABSTRACT

Graduation qualifying bachelor's thesis: 79 pages, 18 images, 2 tables, 5 applications, 13 sources.

Theme: Development of a recommendation system based on content and collaborative filtering.

The object of the research: internet services that include the process of generating recommendations.

The purpose of this work: identification of new patterns and methodological justification of the process of creating a recommendation system in the form of a web application, getting the most relevant recommendations, as well as assessment and comparison of work results; a review of existing approaches to the implementation of recommender systems, based on the studied theoretical material, to develop an algorithm for the formation of a content-based and collaborative recommender system using machine learning; based on the problems identified, proposed improvements in methods.

The subject of the research: methods and models for creating, using personal recommendations (content-based and collaborative filtering).

Results of research: are described possibilities, applications, and evaluation of methods, algorithms of recommender systems. Based on the analysis of results of the system, is made a logical result of observations and suggested improved algorithms.

The conclusion: as a result of researches, the strategy was implemented for providing recommendations, a convenient web application with the built-in strategy that can be used in any modern web service if they need a recommendation mechanism for applying content-based and collaborative filtering.

RECOMMENDATION SYSTEM, MACHINE LEARNING, CONTENT-BASED FILTERING, COLLABORATIVE FILTERING, PYTHON, WEB APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CBF – content-based filtering, фільтрація на основі вмісту, або контентна фільтрація.

CF – collaborative filtering, колаборативна, або спільна фільтрація.

AI – Artificial intelligence, штучний інтелект.

NLP – Natural Language Processing, обробка природної мови.

CSS – Cascading Style Sheets, каскадні таблиці стилів.

HTML – HyperText Markup Languag, мова гіпертекстової розмітки.

URL – Uniform Resource Locator, уніфікований локатор ресурсів або адреса ресурсу.

ВСТУП

Актуальність роботи

Обсяг інформації у всьому світі швидко зростає упродовж тривалого проміжку часу. Щодня люди отримують, сприймають та фільтрують вхідний потік інформації, що надходить із різних джерел: робота, соціальні мережі, електронна пошта, популярні джерела інформації тощо.

За останні роки набули значної популярності інтернет-сервіси, що пропонують товари всіх можливих видів (інтернет-магазини), інформацію на будь-який смак (фільми, новини, форуми, книги, серіали, статті тощо). Стало надзвичайно важно користувачеві орієнтуватися в каталогах товарів, навіть із пошуком товарів, тому як нелегко обрати будь-що за умови великого обсягу інформації. Це створило проблему інформаційного перевантаження. З іншого боку, наука та технології відреагували, розробляючи інструменти фільтрації інформації для розв’язання проблеми. Інтеграція рекомендаційної системи, принципи побудування яких буде розглянуто в цій роботі, стала ефективною стратегією подолання перевантаження.

Системи рекомендацій – це програмні засоби та методи, які надають пропозиції предметів, що, ймовірно, зацікавлять конкретного користувача. Пропозиції стосуються різних процесів прийняття рішень, таких як те, що потрібно купити, яку музику прослуховувати, або які фільми чи серіали переглянути.

Такі системи почали широко використовувати нинішні інтернет-компанії в рамках інтернет-маркетингу. За допомогою прогнозування рекомендацій вони мають на меті збільшити залученість користувачів до конкретного сервісу. Також, у разі розробки рекомендаційної системи з відповідними рекомендаціями, що заслужили довіру користувачів, можна розміщувати серед рекомендацій інші товари, що рекламуються, тобто це зі свого боку сприяє прибутку.

Оскільки якісні рекомендації підтримують зацікавленість клієнтів, рекомендаційні системи є дуже актуальним інструментом для будь-якого бізнесу. Саме тому алгоритми та методи генерації рекомендацій перебувають у постійному

розвитку та стають більш складними, прагнучи підвищити точність та актуальність рекомендацій.

Мета і задачі дослідження

Метою даної бакалаврської роботи є виявлення нових закономірностей та методичне обґрунтування процесу створення системи рекомендацій у вигляді веб-застосунку, отримання найбільш релевантних рекомендацій, а також оцінка та порівняння результатів роботи; огляд наявних підходів до реалізації рекомендаційних систем, на підставі вивченого теоретичного матеріалу, розробити алгоритм формування content-based та collaborative рекомендаційної системи, використовуючи машинне навчання; на основі виявлених проблем запропонувати поліпшення методів.

Отже, основними задачами для досягнення цієї мети є:

- Аналіз підходів та алгоритмів для створення систем рекомендацій.
- Огляд рекомендаційних систем та машинного навчання.
- Проєктування архітектури системи та її поведінки.
- Розробка алгоритмів системи.
- Вивчення технологій аналізу великих масивів інформації.
- Розробка веб-сервісу, що поєднує content-based та collaborative методи.
- Аналіз виконаної роботи та поради щодо поліпшення створених методів.

Об’єкт дослідження є інтернет-сервіси, що включають процес генерації рекомендацій.

Предметом дослідження є методи та моделі створення, користування особистих рекомендацій (контентна та спільна фільтрація).

Методи дослідження

Аналіз, пошук та опрацювання інформації, обробка великих масивів даних, комплексному використанню та поєднанні двох методів фільтрації на основі вмісту та спільної фільтрації, для побудови алгоритму рекомендаційної системи використовується апарат математичної статистики та математичного аналізу, емпіричне порівняння отриманих результатів.

Наукова новизна одержаних результатів

Досліджено можливості рекомендаційних методів та виявлення їх недоліків, що дало змогу запропонувати вдосконалення алгоритмів; виконана програмна реалізація рекомендаційної системи для фільмів з використанням двох методів надання рекомендацій.

Практична цінність одержаних результатів

Реалізовані алгоритми можуть використатись у будь-якому сучасному веб-сервісі, що потребує реалізацію рекомендаційного механізму із застосування підходу фільтрації вмісту та алгоритмів машинного навчання, для того, щоб підвищити прибуток певному бізнесу.

Результати виконаної дипломної роботи впроваджено на підприємстві ТОВ «МІТ».

Особистий внесок студента

Особистий внесок є основні результати:

1. приведена реалізація двох алгоритмів та емпіричне дослідження, які методи варто застосовувати, запропоновано поліпшення методів на основі результатів;
2. розробка комплексу програмного забезпечення – рекомендаційної системи для фільмів.

Апробація результатів випускної кваліфікаційної бакалаврської роботи

Результати бакалаврської роботи представлені на 8-мій Східно-Європейській конференції «Математичні та програмні технології Internet of Everything», що відбулася в м. Києві 14 травня 2021 р.

Публікації

Паралельно з науковими дослідженнями в бакалаврській роботі підготовлено доповідь для 8-мої Східно-Європейської конференції, що надрукована в збірнику узагальнених матеріалів 8-мої Східно-Європейської конференції, секція «Математичні та програмні технології Internet of Everything», що відбулася в м. Києві 14 травня 2021 р.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Кожен день ми зіштовхуємось із вибором і безліччю варіантів. Що подивитись, або купити та прочитати? Тому, аби допомогти користувачеві знайти необхідну інформацію, завзято використовують рекомендаційні системи.

Рекомендаційна система – являє собою підклас систем фільтрації інформації, це програми та сервіси, що намагаються передбачити, які об’єкти (фільми, музика, новини, товари тощо) будуть цікаві користувачеві, спираючись на деяку інформацію (профіль користувача чи опис об’єкту). Ці програми вдосконалили способи взаємодії між сервісом та відвідувачем, оскільки замість того, щоби надавати статичну інформацію, користувач отримує інтерактивні можливості [1].

На рисунку 1.1 зображено загальну архітектуру рекомендаційної системи, яка представлена наступним чином:

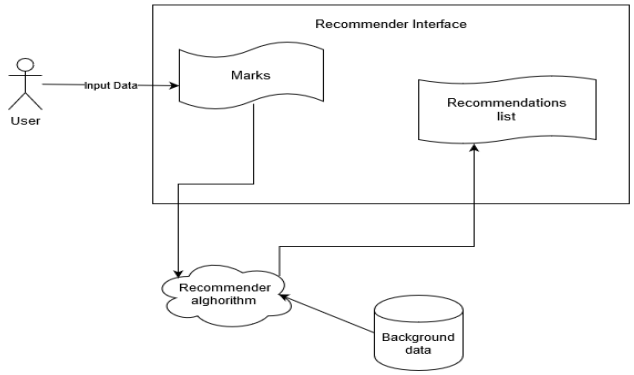


Рис. 1.1. Схема роботи традиційної системи рекомендацій

- дані (background data) – системна інформація про товари сервісу, що збирається;
- вхідні дані (input data) – інформація, яку користувач вносить у систему, щоб отримати рекомендації;
- рекомендаційний алгоритм (recommender algorithm) – алгоритм, що комбінує системну інформацію та вхідні дані для отримання рекомендацій.

Типові системи, у ролі довідкових даних, використовують дані про товар, профілі користувачів, а у ролі вхідних – дії користувача (оцінювання товарів, час, проведений на сторінці тощо). Рекомендаційні системи є чудовою альтернативою пошуковим алгоритмам, оскільки допомагають користувачам виявляти елементи, які в іншому випадку, могли й не знайти.

1.1. Огляд підходів до створення рекомендаційної системи

Рекомендації бувають двох типів: персоналізовані та неперсоналізовані. Прикладом неперсоналізованих – це рекомендувати найпопулярніше, тобто всім одне й теж саме.

Серед підходів персоналізованої рекомендації розрізняють:

- content-based (основані на контенті, тобто вмісту);
- collaborative filtering (колаборативна, або спільна фільтрація);
- гібридну техніку, яка поєднує два попередні методи.

1.1.1. Колаборативна фільтрація

Спільна фільтрація, або колаборативна (collaborative filtering) – це метод прогнозу в рекомендаційних системах, який використовує відомі переваги (оцінки) групи користувачів, або користувача для прогнозування невідомих переваг (оцінок) іншого користувача. За допомогою цього алгоритму будується певна таблиця користувачів, так званої матриці взаємодій користувач-елемент, яка зображена на рисунку 1.2, користувачі групуються за схожістю, та прогнозуються результати для інших користувачів [2].

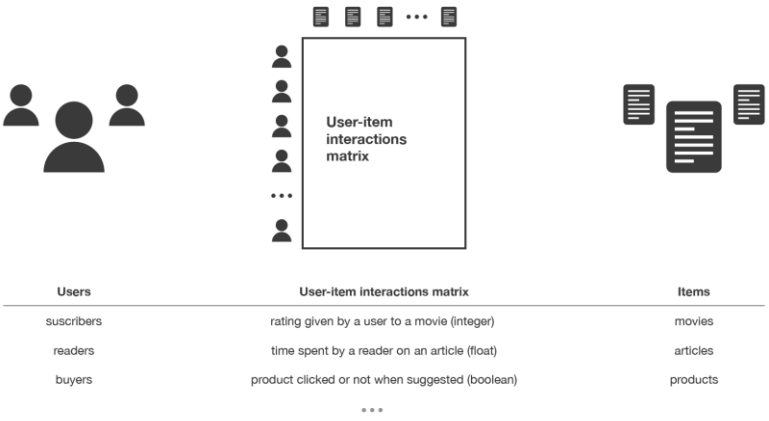


Рис. 1.2. Матриці взаємодій користувач-елемент

Наприклад, під час створення веб-блогу, на якому необхідно запровадити рекомендаційну систему, на основі інформації від багатьох користувачів, які підписуються на блоги й читають їх, можна згрупувати цих користувачів за їх інтересами. На сьогодні дослідники розробили цілий ряд алгоритмів, які можна розділити на дві основні категорії [3]:

– Memory-based, які базуються на аналізі наявних оцінок, анамнестичні методи. Ці алгоритми ґрунтуються на статистичних методах, щоби знайти групу користувачів близьких до цільового користувача. Цей підхід ще називають метод найближчих сусідів: використання попередніх оцінок, зроблених клієнтом, й аналіз оцінок інших користувачів, які мають подібні переваги. Тоді рекомендації (прогноз) для цільового користувача формуються на підставі обчислення якоїсь міри схожості в усіх накопичених даних.

– Model-based, які базуються на аналізі моделі даних, модельні методи. У цьому випадку спочатку за сукупністю оцінок формується описова модель переваг користувачів, товарів і взаємозв'язку між ними, а потім формуються рекомендації на підставі отриманої моделі. Процес формування рекомендацій розбитий на два етапи: ресурсомістке навчання моделі у відкладеному режимі й досить просте обчислення рекомендацій на основі наявної моделі в реальному часі. Однією з

найбільш поширених реалізацій підходу на основі моделі є матрична факторизація.

– Методи, що ґрунтуються на об’єднанні попередніх алгоритмів: гібридні методи.

Три категорії, які згадані вище, також складаються з методів. Розглянемо memory-based методи, які поділяються на:

– Подібності користувачів (user-based) – щоби дати нову рекомендацію користувачеві, метод user-based намагається приблизно ідентифікувати користувачів із найбільш схожим «профілем взаємодії» (найближчими сусідами), щоби запропонувати елементи, які найбільш популярні серед цих сусідів. Цей метод називається «орієнтованим на користувача», оскільки він представляє користувачів на основі їх взаємодії з елементами й оцінює відстані між користувачами.
– Подібності елементів (item-based) – ідея методу полягає в тому, щоби знайти елементи, схожі на ті, з якими користувач уже «позитивно» взаємодіяв. Два елементи вважаються схожими, якщо більшість користувачів, які взаємодіяли з ними обома, робили це однаково. Цей метод називається «орієнтований на предмет», оскільки він представляє елементи, що ґрунтується на взаємодії, які користувачі мали з ним, й оцінює відстані між цими елементами.

Міра схожості відіграє важливу роль у цих методах, адже є основою для порівняння об’єктів. Є кілька способів обрахувати схожість користувачів або елементів: Евклідова відстань, коефіцієнт кореляції Пірсона, манхеттенська відстань та косинусна відстань [4]. Розглянемо більш детально кожен із них.

Евклідова відстань – це простий спосіб розрахунку оцінки подібності. У разі обчислення коефіцієнта подібності за допомогою евклідової відстані предмети оцінювання подаються у вигляді координатних осей. У системі координат розташовуються точки, відповідні перевагам користувачів, на основі яких і визначається коефіцієнт подібності. Розраховується за формулою:

$$d = \|x, y\| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}, \quad (1.1)$$

де d – це відстань, x – координати першого користувача/елемента, y – координати другого користувача/елемента, n – загальна кількість користувачів/елементів, i – номер поточного користувача/елемента.

Кореляція Пірсона – це трохи складніший спосіб визначення подібності інтересів людей. Застосовується для дослідження взаємозв'язку двох змінних, виміряних у метричних шкалах на одній і тій же вибірці. Він дає змогу визначити, наскільки пропорційна мінливість двох змінних. Коефіцієнт кореляції Пірсона характеризує існування лінійного зв'язку між двома величинами. Якщо зв'язок криволінійна, то він не буде працювати. Формула має вигляд:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \quad (1.2)$$

де x – координати першого користувача/елемента, y – координати другого користувача/елемента, n – загальна кількість користувачів/елементів, i – номер поточного користувача/елемента.

Косинусна відстань – це значення відстані між двома векторами внутрішньої площі, який вимірює косинус кута між ними. Також відома, як векторна подібність. Найбільшого поширення в роботах з аналізу даних. Розраховується за формулою:

$$d = cos(\vec{i} \cdot \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|^2 \cdot \|\vec{j}\|^2}, \quad (1.3)$$

де d – відстань між користувачами/елементами, i – вектор першого користувача/елемента, j – вектор другого користувача/елемента.

Манхеттенська відстань – це середня різниця по координатах. У більшості випадків ця міра відстані приводить до таких же результатів, як і для звичайної відстані Евкліда. Однак вплив окремих великих різниць зменшується (через те, що вони не зводяться у квадрат). Манхеттенська відстань обчислюється за формулою:

$$d = \sum_i |x_i - y_i|, \quad (1.4)$$

де x – координати першого користувача/елемента, y – координати другого користувача/елемента, i – номер поточного користувача/елемента.

Колаборативна фільтрація – найпопулярніший сьогодні різновид рекомендаційних систем. Основними перевагами даного підходу для формування рекомендацій: швидка робота алгоритмів, проста реалізація та мала кількість ітерацій [5]. Але головний недолік методу – проблема «холодного старту», тобто для корисних рекомендацій потрібен великий масив даних про переваги користувача, який не доступний на початковому етапі, немає що рекомендувати новим або нетиповим користувачам. Також виникає проблема розрідженої матриці оцінок, оскільки іноді неможливо зробити прогноз.

1.1.2. Фільтрація на основі вмісту (content-based)

Система рекомендацій на основі вмісту намагається рекомендувати товари користувачам на основі їх профілю. Під профілем мається на увазі вподобання та смаки користувача. Він може формуватися на основі кількості натискань на різні елементи або вподобань одного елемента. Іншими словами, рекомендувати елементи, які схожі на ті, які користувачу сподобались у минулому або вивчає зараз.

Зокрема, різні елементи-кандидати порівнюються з елементами, раніше оціненими користувачем, або вподобаними, і рекомендується найбільш відповідні елементи, приклад взаємодії користувач та елемент (фільм, книжка, стаття тощо) зображено на рисунку 1.3.

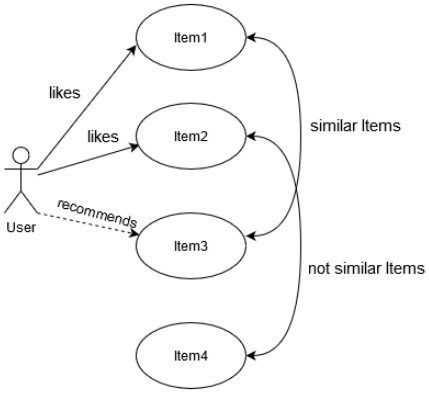


Рис. 1.3. Схема взаємодії користувача й елементів в content-based системі

Основні кроки в даній системі: проаналізувати контент елементів і скласти набір його критеріїв (жанри, теги, слова), дізнатися, які критерії подобається користувачеві, зіставити ці дані та в кінцевому етапі отримати рекомендації. Критерії об'єднують користувачів й елементи в єдиній системі координат, а далі – якщо точка користувача й елемента поруч, то ймовірно елемент сподобається користувачу.

Тобто, чим більше елемент відповідає інтересам користувача, тим вище оцінюється потенційна зацікавленість користувача. Очевидна вимога тут – у всіх товарів/елементів у каталозі має бути опис. Історично предметом content-based рекомендацій частіше були товари з неструктурованим описом: фільми, книги, статті. Такими ознаками можуть бути, наприклад, текстові описи, рецензії, склад акторів та інше. Необхідно враховувати параметри елементів і їх відповідність перевагам користувача. Для цього в рекомендаційних системах використовуються теги (ключові слова), щоб описати об'єкти.

Неструктуровані ознаки описуються типовим для тексту способом – векторами в просторі слів (Vector-Space model). Кожен елемент такого вектору – ознака, потенційно, що характеризує інтерес користувача. Аналогічно, продукт – вектор у тому ж просторі. У міру взаємодії користувача із системою (скажімо, сподобався фільм), векторний опис вподобаного ним елементу об'єднуються (підсумовуються й нормалізуються) у єдиний вектор і формується вектор його інтересів. Далі досить знайти елемент, опис якого найближче до вектору інтересів, тобто розв’язати задачу пошуку *n* найближчих сусідів.

Переваги content-based підходу – це більш точний результат; немає проблеми холодного старту, оскільки рекомендації базуються на моделі об’єкта, а не на попередніх оцінках користувачів.

Недоліками є «затратне» створення моделі (її побудова досить складна), якщо алгоритм складний, то невисока швидкодія (багато обчислень); можлива втрата точності при скороченні параметрів моделі.

1.1.3. Гібридна фільтрація

У всіх різноманітних рекомендаційних систем, розглянутих вище, є свої переваги та недоліки. Деякі системи найбільш ефективні в контекстах, де кількість доступних даних обмежена. Інші, наприклад, колаборативна фільтрація, найкраще працюють у середовищах, де є великі масиви даних. Найчастіше, коли дані різноманітні і достатньо гнучкі, можна скомбінувати рекомендації, отримані декількома способами, тим самим підвищивши якість системи в цілому. Цей тип алгоритмів поєднує в собі підходи колаборативної та content-based фільтрації. Досліджено безліч комбінаторних прийомів, у тому числі:

- Зважені: рекомендацій, отриманих різними методами, присвоюється різну вагу – тобто, деякі рекомендації вважаються кращими, ніж інші.
- Змішані: загальний набір рекомендацій, без явної переваги тих чи інших класів.
- Доповнені: рекомендації від однієї системи використовуються, як введення для наступної, і так по ланцюжку.
- Зміна: вибір випадкового методу.

Отже, гібридний підхід найбільш популярний при розробці рекомендаційних систем для комерційних сайтів тому, що його створено, щоби подолати проблеми колаборативної фільтрації, а також покращити якість прогнозування оцінок конкретної моделі [6].

1. 2. Популярні сервіси, які використовують систему рекомендацій

Рекомендаційні системи використовуються в різних областях і часто розглядаються як генератори списків відтворення для відео- і музичних служб, таких

як Netflix, YouTube і Spotify, рекомендації щодо продуктів, таких як Amazon, або рекомендації щодо вмісту для платформ соціальних медіа, таких як Facebook і Twitter [7]. Оскільки, системи дають можливість утримати клієнта, підвищити залученість аудиторії та вартість кошика.

1.2.1. Netflix – утримання та Amazon – підвищення вартості кошика

Netflix, надає стрімінгові послуги з доступом за підпискою, користувачам пропонуються для перегляду фільми й серіали, що відповідатимуть їхнім інтересам і вподобанням [8]. Імовірність рішення, подивитися конкретне відео з каталогу, оцінюється на основі безлічі чинників:

- Враховуються особливості взаємодії користувача із сервісом (наприклад, історія перегляду та оцінки, виставлені іншим відео).
- Вибір інших користувачів сервісу зі схожими смаками та уподобаннями.
- Характеристики самого фільму або серіалу: жанр, категорія, актори, рік виходу тощо.

Крім інформації про те, що саме користувач дивився на Netflix, також використовує дані про те, як він це робить: у який час доби, на яких пристроях і наскільки довго. Усі ці параметри оцінюються й обробляються. Оскільки більша частина доходів Netflix надходить від періодичної підписки на виставлення рахунків із фіксованою ставкою, найбільша рентабельність інвестицій компанії в системах рекомендацій – утримання.

Розглянемо, як спільна фільтрація працює в реальному житті. Amazon може порекомендувати купити третю книгу до обраних двох, які користувач уже купив, оскільки її купили люди з однаковими інтересами. Як можна побачити, вони не просто сегментують аудиторію і знають, до якої групи ви належите, також

відстежують особисті інтереси конкретного користувача й порівнюють їх з перевагами інших.

Швидке доставлення товарів Amazon та акцент на обслуговуванні клієнтів принесли їм мільйони клієнтів. Системи рекомендацій відіграють роль не тільки у тому, щоб допомогти клієнтам знайти більше необхідного, але й покращують вартість кошика. Amazon не повинен платити більше за доставлення, щоб надіслати удвічі чи втричі більше продуктів, а тому їх норма прибутку збільшується.

1.3. Машинне навчання та обробка природної мови

Машинне навчання є підрозділом штучного інтелекту (AI), що стосується алгоритмів, які значною мірою покладаються на математику та статистику, що дають змогу комп’ютерам навчатися. Типовий випадок використання, коли мало уявляємо, що дані говорять про проблему, яку намагаємося вирішити, наприклад, досить незрозуміло, які фільми подобаються чи ні шанувальникам кіно.

Машинне навчання лежить в основі рекомендаційних систем, а саме: на аналізі даних; дані як навчальний набір, тренування алгоритму та визначення подальших прогнозів. В наступних розділах більш детально розглянуто частини машинного навчання.

Обробка природної мови (Natural Language Processing – NLP) підрозділ інформатики та AI, присвячений тому, як комп'ютери аналізують людську мову. Текст – один із найбільш розповсюджених форм послідовностей даних. У свою чергу, текст можна інтерпретувати і як послідовність символів, і як послідовність слів, але частіше обробляється на рівні слів. У NLP (обробка природної мови) під текстом розуміється упорядкована послідовність токенів, де токени – це символи, слова або речення. Набір усіх текстів, які є в наявності в задачах NLP, заведено називати корпусом, а набір усіх унікальних токенів – словником.

Для зменшення розміру словника, скорочення кількості обчислень і поліпшення якості векторних уявлень текстів використовують методи попередньої обробки текстів. Попередня обробка тексту традиційно є важливим кроком, адже має на меті перетворити текст у більш зручну форму, щоб алгоритми машинного навчання могли працювати краще.

Для цього необхідно виконати над текстом наступні операції:

- Токенізація – розбиття довгих ділянок тексту на більш дрібні (абзаци, речення, слова), найперший етап обробки тексту.
- Нормалізація – приведення кожного окремого слова до його початкової форми (єдиний реєстр слів, відсутність знаків пунктуації, розшифровані скорочення, словесне написання чисел тощо).
- Стеммізація – приведення слова до його кореня через усунення придатків, тобто суфікса, приставки, закінчення.
- Лематизації – приведення слова до смислової канонічної форми слова (інфінітив для дієслова, називний відмінок однини – для іменників і прикметників).
- Чистка – видалення стоп-слів, які не несуть смислового навантаження, а саме: артиклі, сполучники, прийменники тощо.

По завершенні попередньої обробки, текст стає придатним для його переведення в числову форму, тобто векторизацію (перетворення в числові вектори), щоби далі продовжити витяг ознак. Для такої трансформації використовують спеціальні моделі, найбільш популярними з яких є:

- Word2Vec – спочатку створюється словник, «навчаючись» на вхідних текстових даних, а потім обчислюється векторне подання слів, засноване на контекстної близькості. При цьому слова, що зустрічаються в тексті поруч, у векторному поданні матимуть близькі числові координати. Слово "чоловік" належить до слова "жінка" так само як слово "дядько" до слова "тьотя", в інших моделях домогтися такого ж співвідношення векторів можна тільки за допомогою спеціальних хитрощів. Тобто варто

розуміти, що ця модель, звичайно ж, не володіє розумінням слів, а просто намагається розмістити вектори так, щоб слова, що вживаються в загальному контексті, розміщувалися недалеко один від одного.

- TF-IDF (Term Frequency – Inverse Document Frequency, частота слова – зворотна частота документа) – підраховується входження кожного слова в документі, оцінюється важливість кожного слова й обчислюється оцінка для цього документа. Суть методу, якщо слово часто згадується в даному тексті і якщо це не стоп-слово, то, швидше за все, воно важливе. Але також, якщо якесь слово рідко згадується в інших документах, а в даному документі воно зустрічається, то воно, теж важливо для даного документа, тому що по цьому слову можна відрізнити даний текст від усіх інших.
- Count Vectorizer – модель не враховує граматику або порядок слів і потрібна для визначення кількості входжень окремих слів в аналізований текст. На практиці реалізується в такий спосіб: створюється вектор довжиною в словник, для кожного слова рахується кількість входжень у текст і це число підставляється на відповідну позицію в векторі. Однак, при цьому втрачається порядок слів у тексті, а значить, після векторизації речення, наприклад, «i have no cats» і «no, i have cats» будуть ідентичні, але протилежні за змістом.

1.4. Висновки до розділу 1

У підсумку, проаналізовані основні підходи до створення рекомендацій: основані на контенті, колаборативну та гібридну фільтрацію. Описані плюси та мінуси кожного, а також більш детально переглянуті методи та алгоритми для подальшої розробки системи. При огляді способів отриманні векторизації, виникло

питання вибору методу, адже кожен залежить від поставленого завдання. Зіставивши суть використання Word2Vec, TF-IDF та Count Vectorizer у попередніх підрозділах, вирішено використовувати останню модель у розроблюваному підході основаному на контенті. Тому що модель Count Vectorizer швидка у використанні, проста в реалізації, та оскільки об’єм тексту і слів буде невеликий, то метод цілком підходить для вирішення завдання.

РОЗДІЛ 2
ІМПЛЕМЕНТАЦІЯ МАТЕМАТИЧНИХ ОСНОВ В АЛГОРИТМИ

2.1. Математичний опис створення content та collaborative filtering

При роботі з великим об’ємом даних фільмів у форматі .csv, потрібно правильно характеризувати їх. Використовуватимемо векторну модель (term vector model), оскільки – це алгебраїчна модель для подання текстових документів (чи будь-яких об'єктів, у загальному), у ролі векторів ідентифікаторів, таких як, наприклад, індексних термінів, або тегів (terms), з якими найкраще працювати при порівняннях [9]. Вектором є величина, яка характеризується, як числовим значенням, так і напрямом у просторі й має вигляд:

$$d_j = (w_{1,j}, w_{2,j} \dots, w_{i,j}), \tag{2.1}$$

де d_j — векторне уявлення j-го документа, w_{ij} — вага i-го терміну в j-му документі, n — загальна кількість різних термінів.

Велика кількість векторів буде представлена у вигляді матриці, тобто спеціального вигляду прямокутна таблиця, що буде складатися з рядків – фільмів та стовпців – жанру, акторів, режисеру, опису кіно для контентної фільтрації, а для колаборативних методів з триплетів (фільми – стовпці, користувачі – рядки, в комірках матриці розташовані оцінки). Розмір матриці задають стовпці та рядки, вона має вигляд:

$$\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & a_{m1} & a_{m1} \end{bmatrix}, \tag{2.2}$$

де $a_{i,j}$ – елемент матриці; i – номер рядка; j – номер стовпця.

Документом може бути будь-який об’єкт, який потрібно змоделювати, наприклад, люди можуть мати вік, вагу, ріст тощо. У нашому випадку в контентній фільтрації – це буде жанр, актори, режисер, опис фільму, які будуть об’єднані надалі в один вектор, але попередньо переведені до порядку. Колаборативній фільтрації

безліч триплетів (користувач, фільми, оцінка) оцінок формують розріджену матрицю, яка називається матрицею оцінок, пари (користувач, фільм), в яких користувачі не віддали перевагу фільму, є невідомими значеннями цієї матриці Векторні операції можна використовувати для порівняння документів із запитами. Тобто між тим, що сподобалось користувачеві та термінами, тобто нашими ключовими словами. Переваги векторної моделі – оскільки модель проста, можна розраховувати подібність між запитом і документом нескінчену кількість, модель дозволяє знайти збіги між елементами (фільмами та користувачами) [10].

2.2. Обґрунтування вибору мір схожості

Для того, щоб обчислити схожість між двома об'єктами, використовують міру схожості, як зазначено в попередніх розділах, вона відіграє важливу роль. Адже від того, як розрахувати подібність двох об’єктів, залежить надалі генерація рекомендацій.

Двома популярнішими мірами є косинусна відстань та кореляція Пірсона. Зважаючи на дослідження емпіричного порівняння індексів локальної структурної подібності для рекомендаційних систем [11], де проводять паралель результатів мір, а саме: Салтона, Жаккарда, Серенсена, Адаміча-Адара, розподіл ресурсів, косинусного індексу та коефіцієнта кореляції Пірсона відповідно (на наборі даних із сайту MovieLens). У такий спосіб, порівняння результатів, виміряні точністю, різноманітністю та популярністю, продемонстрували, що косинусна відстань та кореляція Пірсона дійсно можуть дати хороші результати.

Для контентної фільтрації відповідність між двома фільмами з ключовими словами, використовуватимемо косинус кута між векторами. Який розраховується за формулою:

$$\cos\theta = \frac{d_i \cdot q_j}{\|d_i\|\|q_j\|}, \tag{2.3}$$

де $d_i \cdot q_j$– перетин, тобто скалярний добуток i та j векторів ключових слів фільмів, а $\|d_i\|\|q_j\|$ – нормалі векторів, які обчислюються за формулою:

$$\|q\| = \sqrt{\sum_{j=1}^{n} q_j^2}\,, \tag{2.4}$$

де q_j – це j вектор ключових слів фільму. Значення косинуса кута, який перебуває в діапазоні від -1 до 1, вказує на проміжну схожість чи несхожість векторів, де -1 абсолютно не схожі, а 1 абсолютно однакові вектори, а при 90° мають схожість 0 [12]. Тобто два вектори «подібні», у нашому випадку два фільми подібні, якщо вони паралельні та максимально «відрізняються», якщо вони ортогональні (перпендикулярні).

В колаборативній фільтрації реалізовуватимемо відразу два методи засновані на пам’яті: item-based та user-based. З огляду на дослідження, для цих методів можливе застосування, як косинус кута, так і кореляцію Пірсона. Тому для user-based використовуватимемо косинус кута (див. формулу 2.3), де $d_i \cdot q_j$ – це перетин між користувачами i та j. А для item-based кореляцію Пірсона, яка розраховується за формулою:

$$w_{a,b} = \frac{\sum_{u \in U}(r_{i,a} - \overline{r_i})(r_{j,b} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{i,a} - \overline{r_i})^2}\sqrt{\sum_{u \in U}(r_{j,b} - \overline{r_j})^2}}\,, \tag{2.5}$$

де U – множина користувачі, $r_{i,a}$ – координати першого користувача, $r_{j,b}$ – координати другого користувача, які оцінили елементи i та j. Надалі передбачення рекомендацій базується на основі отриманих обчислень мір кожного користувача/фільму близьких до користувача, який ввів відповідний фільм та оцінку. Тобто прогнозування може бути обчислено, як зважений скалярний добуток, як для item, так і для user-based розраховується за формулою:

$$p_{u,i} = \sum_f u_f\, w_f i_f\,, \tag{2.6}$$

де u – вектор інтересу користувача (назва фільму), i – вектор оцінки, w – вектор релевантного елемента/користувача. Вектор оцінки може додатково бути нормалізованим, шляхом віднімання будь-якої константи, нехай константна дорівнюватиме 2,5.

2.3. Висновки до розділу 2

Описані ключові моменти математичної частини контентної та колаборативної фільтрації. У результаті обрано косинус кута та кореляцію Пірсона, таким способом зрозуміти чи практично використовувати широко використовувані метрики та проаналізувати отримані результати. Адже як такого ідеальної, або найкращої оцінки двох об’єктів немає, є лише рекомендації до застосування того чи іншого методу.

РОЗДІЛ 3
ПРОЄКТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1. Вимоги до програмного забезпечення

Функціональні вимоги, «що» мусить робити система:

1) введення першого фільму для content-based рекомендації, введення другого фільму та його оцінка для collaborative filtering (фільми можуть збігатися);
2) перевірка фільмів на коректність введення;
3) вивід повідомлення про помилку;
4) вивід трьох різних рекомендацій фільмів;
5) надавати інформацію про те, що це за системи та який набір даних використовується;

Нефункціональні вимоги, «яка» повинна бути система:

1) програма повинна мати інтерфейс користувача:
    а) виконання надійне (без помилок);
    б) доступність інтерфейсу;
    в) ефективність;
    г) прозорий та виразний веб-інтерфейс англійською мовою, який повинен мати:
        - «головну сторінку», де повинна бути: назва системи; поле для вводу двох назв фільмів та оцінки другого; кнопку «Submit»; меню, де розміщуватиметься інформація про систему та посилання на список усіх фільмів;
        - «сторінка про помилку», де: інформація про помилку; повернення на «головну сторінку»;

– «сторінка рекомендацій», де: назва системи; 10 кращих рекомендацій фільмів у вигляді списку для content-based; 10 кращих рекомендацій фільмів у вигляді списку для collaborative filtering (item and user based); меню, де інформація про систему та посилання на список усіх фільмів;

2) введення назв фільмів англійською мовою;
3) простота в експлуатації.

3.2. Діаграми використання, взаємодії та компонентів рекомендаційної системи

На основі вимог до веб-застосунку для більшого розуміння системи створено діаграму варіантів використання – це графічний засіб специфікування вимог, яка зображена на рисунку 3.1.

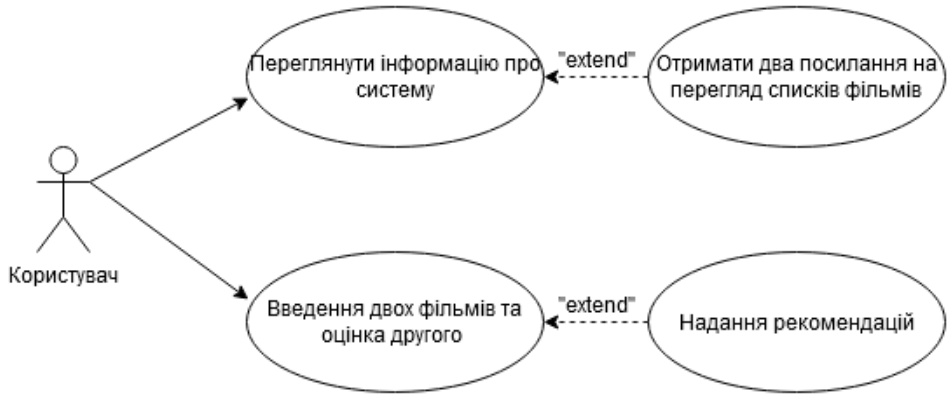


Рис. 3.1. Діаграма використання системи рекомендацій

Можна побачити на рисунку 3.1 суб’єкт, а саме користувач, який може взаємодіяти із системою, для користувача отримано варіанти використання: продивить інформацію про систему, а далі отримати посилання на перегляд фільмів;

ввести два фільми та оцінка другого фільму, користувач вводить фільми та отримує рекомендацію.

Так, як це буде веб-застосунок, то використовувати клієнт-серверну архітектуру, яка працює на сторінці веб-сервера, WSGI конфігурація вбудована у фреймворк, та використовує веб-браузер (клієнт), взаємодія відбувається з протоколом HTTP, що проілюстровано на рисунку 3.2.

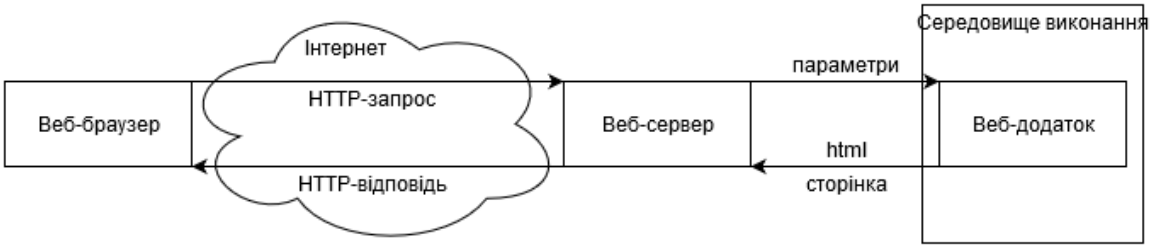


Рис. 3.2. Загальна клієнт-серверна схема

Такий вибір зумовлений тим, що він є простий у використанні, стійкість до атак та має швидку реакція на дії користувача. Для того, щоби зрозуміти, як користувач буде взаємодіяти із системою, створено діаграму взаємодії на рисунку 3.3, яка використовується для взаємодії між об’єктами програмного забезпечення, для прецеденту Надання рекомендації фільмів.

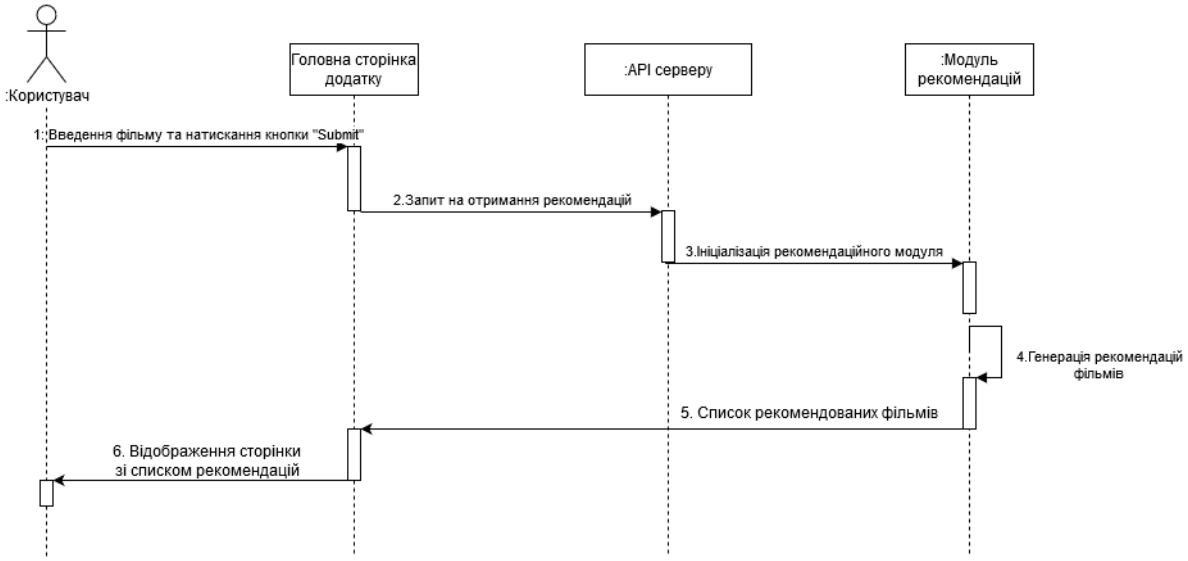


Рис. 3.3. Діаграма взаємодії користувача із системою рекомендацій

Вхідною інформацією буде – назва двох фільмів та оцінка, вихідною – список фільмів з 10 найкращих схожих фільмів до першого введеного та 20 рекомендацій для другого. Можна дійти висновку, що взаємодія користувача з системою відбувається покроково та послідовно. Отже, описано, як повинна поводитись програма, потрібно зрозуміти фізичне представлення веб-застосунку, тобто які компоненти повинні бути реалізовані для створення веб-сервісу рекомендацій. Для цього створимо діаграму компонентів, що має програні модулі, які розміщені в пакетах, як представлено на рисунку 3.4.

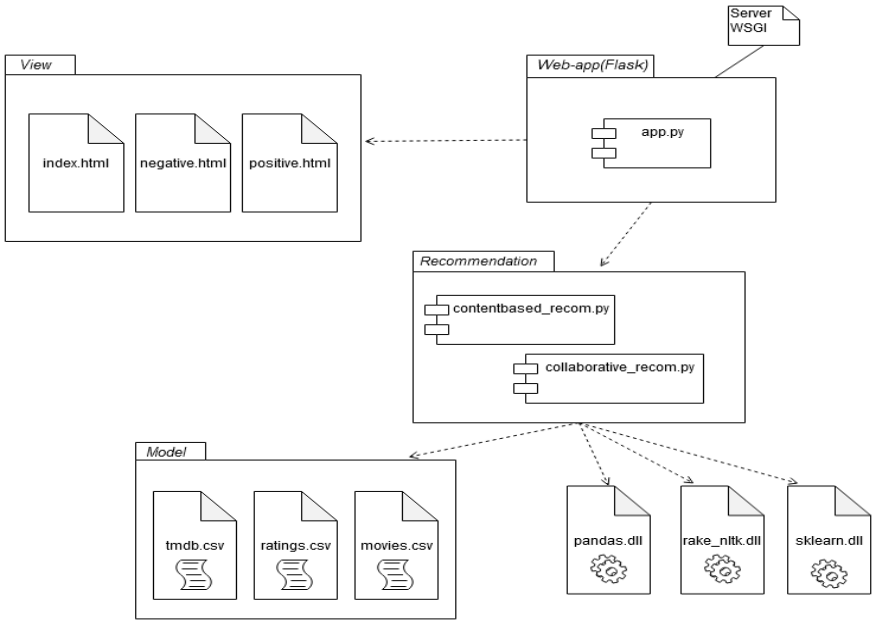


Рис. 3.4. Діаграма компонентів веб-застосунку рекомендацій

Тобто маємо:

- головний модуль app.py, який буде генерувати веб-застосунок, в якому імпортований Flask і створений екземпляр класу Flask, що і буде нашим WSGI-додатком; функція, яка генерує URL-адрес і повертає відповідну .html сторінку, а також викликає функцію recommendations з модуля contentbased_recom.py та функції getRecomForItems, getRecomForUsers з модуля collaborative_recom.py;
- папка Recommendation, де 2 модулі рекомендацій, а саме: contentbased_recom.py, у якому імпортовані 3 бібліотеки, функція

зчитування даних, перетворення цих даних, тобто попередня обробка, функція get_similarity(), де Count Vectorizer та функція Cosine Similarity, а також функція recommendations(); collaborative_recom.py, де імпортовані 3 бібліотеки, функція data_processing(), що виконує попередню обробку та розрахунок матриці подібності, get_similar_movies() – розрахунок подібності для фільмів та get_similar_users() – для користувачів, дві функції рекомендацій getRecomForItems(), getRecomForUsers().

- папка Model, де будуть наші дані про фільми у форматі .csv;
- папка View – 3 веб-сторінки, які складаються зі структури сторінки (html) та зовнішнього вигляду (css); index.html – матиме поле для вводу, кнопку «Submit», кнопку меню, опис системи; positive.html – кнопку меню, опис системи, список рекомендацій; negative.html – інформацію про проблему та кнопку повернення назад.

3.3. Алгоритми генерації рекомендації для content та collaborative filtering

Найважливіша частина системи рекомендацій – алгоритм. У роботі використовується два методи CBF та CF (item та user based). Для CBF обрано набір даних, що містить 250 кращих фільмів IMDB англійською мовою, бо набір має відповідні елементи для контентної фільтрації. Таблиця містить 250 фільмів і 38 стовпців. Однак, потрібно, щоби рекомендації ґрунтувалися тільки на режисера фільму, головних акторів, жанрі й сюжеті, адже саме вони несуть головну інформацію про фільм, тому це єдині стовпці, які будемо розглядати в моделюванні.

Основні кроки в цьому алгоритмі – це проаналізувати дані, які в нас будуть в .csv форматі, створити набір критеріїв (жанрів, опис фільму, режисеру), об’єднати ці критерії та знайти ключові слова, порівняти обраний фільм та набір даних, отримати рекомендацію. Більш детальний опис алгоритму описано в Додатку А.1. Дістаємо

тільки один стовпець для кожного фільму, який містить усі вищезазначені характеристики разом. Для цього потрібно виконати векторизацію для обчислення косинуса та отримати рекомендацій.

Для CF обрано інші дані, адже знайти відповідний набір, аби в ньому були присутні й користувачі, і рейтинги, і теги виявилось складним завданням. Набір даних (ml-latest-small) від MovieLens, що створені 610 користувачами в період з 29 березня 1996 року по 24 вересня 2018 року, містить 100836 рейтингів та 9742 фільми. Основні кроки – об'єднання файлів, отримання стовпців назв фільмів, id користувача, рейтинг; в обох випадках необхідно створення user-item матрицю, яка матиме стовпець id користувача, рядки – фільми, в комірках матриці буде записана інформація про оцінку фільму *m* користувачем *n*. Після побудови цієї матриці, на її основі необхідно розрахувати дві нові матриці з коефіцієнтами подібності для користувачів (косинус кута) і для фільмів (кореляція Пірсона).

Рекомендації для item-based: отримуємо введений фільм, знаходимо користувачів, яким подобається цей фільм, дивимося на фільми, які подобаються знайденим користувачам, виводимо 10 список кращих фільмів за зменшенням. Рекомендації для user-based: беремо вихідного користувача, знаходимо групу користувачів, яка максимально схожа на нього (ґрунтуючись на оцінках), дізнаємося, які фільми сподобалися цій групі, виводимо список 10 кращих фільмів за зменшенням. Детальний опис алгоритмів описано в Додатку А.2.

Через те, що у нас буде веб-застосунок, опишемо його роботу в цілому. Маємо три сторінки веб-застосунку: це «Головна сторінка», «Сторінка помилки» та «Рекомендацій». Якщо користувач обирає ввести перший фільм, другий фільм і його оцінку, натискає на кнопку «Submit», він переходить на сторінку рекомендацій, якщо – введені фільми та оцінка некоректні (неправильна назва, введений з помилками, відсутній в списку, оцінка не в межах 1-5), користувач перейде на сторінку про помилку, де можна буде повернути назад, а саме на головну сторінку.

Якщо користувач бажає отримати інформацію про систему, то переходить до меню, де міститься опис системи та посилання на список двох наборів даних фільмів,

які використовує система, в іншому випадку закриває меню та повертається на головну сторінку, або зовсім виходить з системи.

3.4. Висновки до розділу 3

У цьому розділі виконані основні кроки проєктування системи, у результаті отримано: визначення внутрішніх властивостей системи, деталізації її зовнішніх (видимих) властивостей, обрано подання внутрішніх даних; розробка основних алгоритмів; описана поведінка системи та фізичне представлення веб-застосунку. Що дає змогу приступити до реалізації самого проєкту.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ РЕКОМЕНДАЦІЙ ФІЛЬМІВ

Після аналізу, визначення вимог та проєктування настає час їх впровадження. У цьому розділі описані основні аспекти, використовувані засоби та застосування рішень.

4.1. Вибір мови програмування, бібліотек та фреймворку

Оскільки робота зв’язана з машинним навчанням, то обрано мову Python та середовище розробки Visual Studio Code. Адже Python використовується для машинного навчання майже всіма розробниками. По-перше, це проста мова для розуміння, що пропонує надійний код. Тож простота Python допомагає впоратися зі складними алгоритмами, крім того, це економить час розробки. По-друге, чудова бібліотечна екосистема є однією з головних причин, чому Python є кращим для машинного навчання. Бібліотеки допомагають і скорочують час розробки. В роботі використовуються три бібліотеки, а саме:

- Pandas – пакет, що забезпечує швидкі, гнучкі та виразні структури даних, розроблені для того, щоби зробити роботу з «реляційними» або «міченими» даними одночасно легко та інтуїтивно [13].
- Scikit-learn – модуль Python, що інтегрує класичні алгоритми машинного навчання і тісно пов'язаний з науковими пакетами (numpy, scipy, matplotlib), спрямований на просте та ефективне розв'язання проблем машинного навчання.
- Rake_nltk – алгоритм швидкого автоматичного вилучення ключових слів, шляхом аналізу частоти появи слова та його співзвучності з іншими словами в тексті.

Вони дають змогу виконувати матричні операції (табличні) дуже швидко (швидше, ніж будь-яка самописна ітерація).

Щоб створити веб-сторінку обрано HTML і CSS. Оскільки вони є актуальними інструментами для веб-дизайну, адже є простими у використанні та розуміння, навіть, якщо до цього не працювали з HTML та CSS.

Фреймворки полегшують повторне використання коду для звичайних операцій HTTP. Власне, ці фреймворки інкапсулюють роботу. Для створення веб-сервісу вирішено обрати веб-фреймворк Flask, адже забезпечує простоту, гнучкість, має невелике ядро і легко розширюваний. В основі своєї роботи Flask використовує WSGI, який забезпечує маршрутизацію URL-адрес та обробку запиту/відповіді. Тобто сервер WSGI просто викликає об'єкт, що викликається, у програмі. Об'єкт запиту є глобальним, тому можете отримати доступ до нього набагато простіше, URL-адреси у фреймворку, як правило, визначаються разом із представленням (за допомогою декоратора).

4.2. Аналіз та попередня обробка наборів даних

Для того, щоби реалізувати рекомендації фільмів на основі content-based методу, взято дані з сайту IMDB топ кращих 250 фільмів. Працюватимемо з Pandas DataFrame для зручності, зі свого боку – це представлення у вигляді двовимірних, з розміром, табличних даних. Зчитуємо дані з файлу за допомоги pd.read_csv(). Стало зрозуміло, що ця таблиця містить 250 фільмів та 38 стовпців. Для рекомендацій всі стовпці нам не потрібні, адже основна інформація для характеристики фільмів знаходиться в: Назві фільму, Жанрі, Режисері, Акторів, Сюжету. Спробуємо подивитись, в якому році найбільша кількість фільмів та з якого по який рік присутні фільми. Поглянувши на рисунок 4.1 можна побачити, що фільми йдуть з 1921 року до 2017, найбільше фільмів з 2010 до 2017 року.

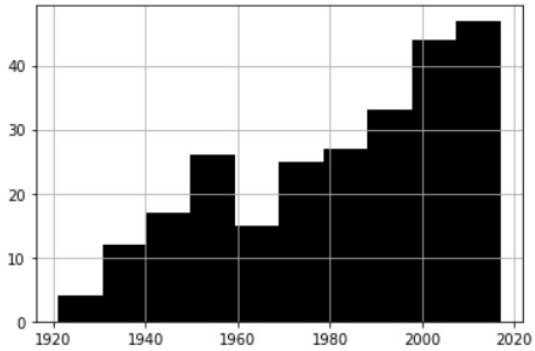


Рис. 4.1. Діаграма відношення кількості фільмів до року випуску

Наступний крок, обираємо відповідно рядки, які потрібно та приводимо їх до порядку, як це описано в Додатку А, на рисунку 4.2 зображена таблиця до перетворень.

	Title	Genre	Director	Actors	Plot
0	The Shawshank Redemption	Crime, Drama	Frank Darabont	Tim Robbins, Morgan Freeman, Bob Gunton, Willi...	Two imprisoned men bond over a number of years...
1	The Godfather	Crime, Drama	Francis Ford Coppola	Marlon Brando, Al Pacino, James Caan, Richard ...	The aging patriarch of an organized crime dyna...
2	The Godfather: Part II	Crime, Drama	Francis Ford Coppola	Al Pacino, Robert Duvall, Diane Keaton, Robert...	The early life and career of Vito Corleone in ...
3	The Dark Knight	Action, Crime, Drama	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart, M...	When the menace known as the Joker emerges fro...
4	12 Angry Men	Crime, Drama	Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G....	A jury holdout attempts to prevent a miscarria...

Рис. 4.2. Таблиця даних до перетворень

Попередню обробку тексту проводимо наступним чином:

– виявлено приємну особливість у пакеті nltk, яка дозволяє витягувати ключові слова з тексту, у нашому випадку використовуємо її в описі фільму, витяг зображено на рисунку 4.3.

Plot
Two imprisoned men bond over a number of years, finding ...
The aging patriarch of an organized crime dynasty transf...
The early life and career of Vito Corleone in 1920s New ...
When the menace known as the Joker emerges from his myst...
A jury holdout attempts to prevent a miscarriage of just...

↓

Key_words
[finding, solace, years, acts, number, eventual, redempt...
[aging, patriarch, organized, crime, dynasty, transfers,...
[portrayed, 1920s, new, york, family, crime, syndicate, ...
[mysterious, past, gotham, ability, dark, knight, must, ...
[miscarriage, jury, holdout, attempts, colleagues, justi...

Рис. 4.3. Витяг ключових слів

– переводимо «Жанр», «Режисер», «Актори» і «Ключові слова» в нижній регістр, щоб уникнути дублювання;
– об'єднуємо всі імена і прізвища в одне унікальне слово в «Режисер» і «Актори»;
– після попередньої обробки «Жанр», «Режисер», «Актори» і «Ключові слова» об'єднуємо в новий стовпець «Bag of words» та перепризначаємо індекс на стовпець назви фільмів.

Усі попередні кроки зображені покроково на рисунку 4.4.

Рис. 4.4. Таблиця даних після перетворень

Програмний код попередньої обробки можна переглянути в Додатку Б.1. Після цього дані готові для маніпуляції надалі для функцій get_similarity і recommendations.

Для CF взято набір даних із сайту MovieLens. 943 користувачів дали оцінки 1682 фільмам, усього 100 000 рейтингів. Середній рейтинг – 3,52 за шкалою від 1,0 до 5,0 зі стандартним відхиленням 1,12, медіана рейтингу – 4,0. Візуалізуємо розподіл рейтингів, що можна побачити на рисунку 4.5.

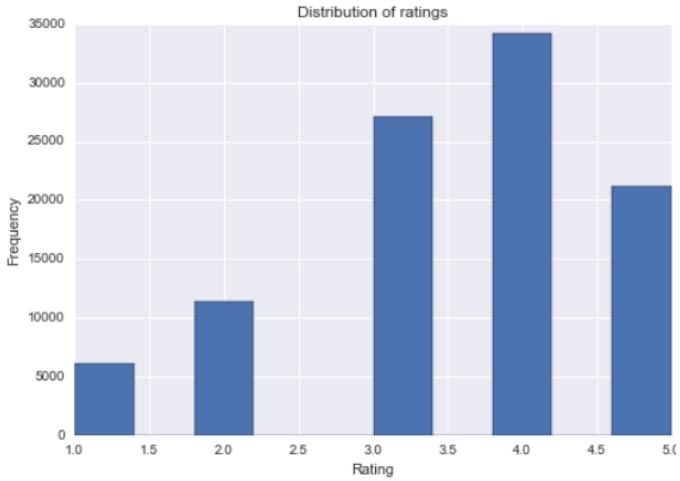


Рис. 4.5. Розподіл рейтингів

Дізнаємося скільки фільмів оцінили люди в середньому. Здається, більшість користувачів оцінили менше ніж 50 фільмів (див. рис. 4.6).

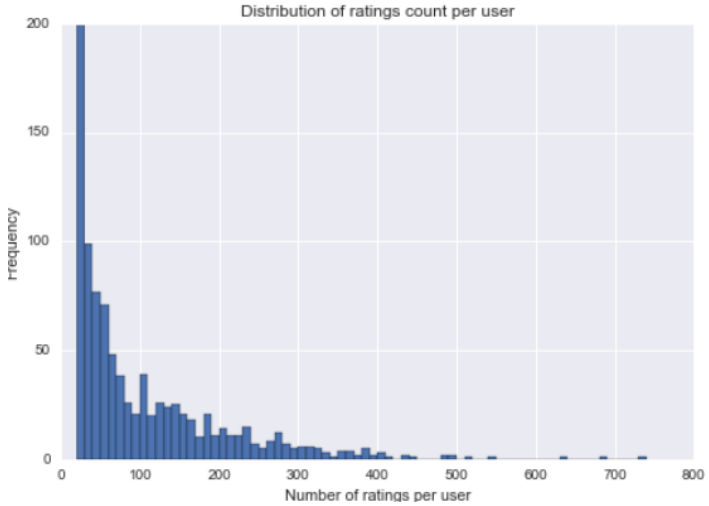


Рис. 4.6. Розподіл рейтингів

Для CF робота з даними набагато простіше, ми зчитуємо два файли ratings.csv та movies.csv, об’єднуємо ці файли, видаляємо непотрібні стовпці, залишаємо лише movieId, title, userId, rating та створюємо відповідний DataFrame ratings (див. рис. 4.7).

	movieId	title	userId	rating
0	1	Toy Story (1995)	1	4.0
1	1	Toy Story (1995)	5	4.0
2	1	Toy Story (1995)	7	4.5
3	1	Toy Story (1995)	15	2.5
4	1	Toy Story (1995)	17	4.5

Рис. 4.7. Таблиця даних після обробки

Дізнаємося унікальних користувачів 610 і фільмів 9724. Лістинг попередньої обробки можна переглянути в Додатку Б.2, наразі дані готові до подальшого маніпулювання.

4.3. Реалізація функцій рекомендацій

Для реалізації функції рекомендації CBF, подивимось на рисунку 4.4, приведено дані таблиці до порядку, тепер потрібно зробити векторизація з допомоги get_similarity(). Count Vectorizer – простий лічильник частот для кожного слова в стовпці Bag of words. Після того, як з'явилась матриця, що містить кількість усіх слів у фільмі, можна застосувати функцію співвідношення cosine similarity для порівняння подібності між фільмами. На рисунку 4.8 зображена матриця, ми бачимо одиниці по діагоналі, оскільки, звичайно ж, кожен фільм ідентичний самому собі.

```
[[1.         0.15789474 0.13764944 ... 0.05263158 0.05263158 0.05564149]
 [0.15789474 1.         0.36706517 ... 0.05263158 0.05263158 0.05564149]
 [0.13764944 0.36706517 1.         ... 0.04588315 0.04588315 0.04850713]
 ...
 [0.05263158 0.05263158 0.04588315 ... 1.         0.05263158 0.05564149]
 [0.05263158 0.05263158 0.04588315 ... 0.05263158 1.         0.05564149]
 [0.05564149 0.05564149 0.04850713 ... 0.05564149 0.05564149 1.        ]]
```

Рис. 4.8. Матриця подібності

Матриця також симетрична, тому що схожість між A і B така сама, як схожість між B і A. Останнім кроком є створення функції рекомендації, яка приймає на вхід назву фільму й повертає 10 найкращих подібних фільмів. Для цього створено просту серію назв фільмів з числовими індексами, щоб зіставити індекси з матриці подібності з фактичними назвами фільмів. Ця функція буде зіставляти назву вхідного фільму з відповідним індексом матриці подібності та витягувати рядок значень, відповідні індекси, в порядку спадання. Кращі 10 схожих фільмів можна знайти, витягуючи 11 верхніх значень і згодом відкидаючи перший індекс (щоби функція не

повертала ту ж назву фільму). Більш детальну реалізацію написаною мовою Python можна подивитись в Додатку Б.1.

CF реалізуємо відразу два методи item-based і user-based. Приведена таблиця на рисунку 4.7 готова для побудови user-item матриці. Вона є вихідною матрицею для двох методів, колонкам даної матриці відповідає об'єкт, в нашому випадку фільм, рядку відповідає користувач, в комірках матриці розташовані рейтинги зображені на рисунку 4.9.

userId	'burbs…	(500) …	10 Clo…	10 Thi…	10,000…
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Рис. 4.9. User-item матриця

Для зменшення шуму в рекомендаціях і збільшення їх точності використовуємо тільки ті фільми, у яких є понад 10 користувачів, що оцінили його. Основною матрицею для item-based підходу є матриця item-item. Рядках і стовпцях цієї матриці відповідають об'єкти (фільми). У комірках зберігається значення схожості фільмів. Для визначення схожості використовуємо кореляцію Пірсона, в створеній функції get_similar_movies() виконуємо item_similarity_df = user_ratings.corr(method='pearson') та зберігаємо ці дані до item_similarity_df.csv файлу (див. рис. 4.10).

	title	'burbs…	(500) …	10 Clo…	10 Thi…
0	'burbs, …	0	0.063116…	-0.02376…	0.143481…
1	(500) Da…	0.063116…	0	0.142471…	0.273989…
2	10 Clove…	-0.02376…	0.142471…	0	-0.00579…
3	10 Thing…	0.143481…	0.273989…	-0.00579…	0

Рис. 4.10. Item-item матриця для item-based підходу

У item-item матриці всі значення по діагоналі дорівнюють 1 (об'єкт на 100% схожий на себе), тому зануляємо одиниці, щоб в рекомендаціях не було того ж самого введеного фільму. Коли у нас є матриця item-item, можемо побудувати

персоналізовані рекомендації для користувача А. Створюємо функцію def getRecomForItems(movie, rating), що приймає назву фільму і рейтинг, як вхідні дані і повертає 10 схожих фільмів. Перемножуємо матрицю item-item і вектор рейтингу користувача А, віднімаючи константу 2,5, записуємо в дата фрейм similar_movies; сумуємо та відсортовуємо в порядку убування значень отриманого списку фільмів, зберігаємо список фільмів до масиву all_recommend, беремо тільки перші 10 фільмів та повертаємо їх.

Для user-based підходу також використовуємо матрицю user-item. Для кожного користувача необхідно отримати найбільш схожих користувачів, як писалося вище, для міри близькості будемо використовувати косинусну відстань. У функції get_similar_users() потрібно транспонувати матрицю user-item (має назву user_ratings), викликаючи атрибут .T та обчислюємо косинусну відстань, тобто у комірках будуть значення близькості кожного користувача, індекс – користувач, стовпці – користувач та приведемо матрицю до дата фрейму (див. рис. 4.11).

	0	1	2	3
0	1	0.099735…	0	0.180400…
1	0.099735…	1	0.173842…	0.326121…
2	0	0.173842…	1	0.036934…
3	0.180400…	0.326121…	0.036934…	1

Рис. 4.11. Матриця подібності user-user

Також зануляємо діагональ нашої таблиці. Далі створюємо функцію def getRecomForUsers(movie_name, ratings), що приймає назву фільму і рейтинг, у ролі вхідних даних і повертає 10 схожих фільмів. Перемножуємо матрицю user-user і введений фільм, вектор рейтингу користувача А, віднімаючи константу 2,5, записуємо до таблиці similar_score, вибираємо підмножину користувачів з най високою схожістю з користувачем А, сортуємо та вибираємо тільки 10 користувачів, зберігаємо до матриці sim та виводимо список 10 кращих схожих фільмів.

4.4. Реалізація модуля Flask та веб-сторінок

Для початку імпортували клас Flask та модулі рекомендацій. Екземпляр цього класу й буде WSGI-додатком. Наступним кроком є створення екземпляру цього класу. Перший аргумент – це ім'я модуля або пакета програми (app = flask.Flask(__name__, template_folder='templates')). Далі, використовуємо декоратор @app.route(), щоби сказати Flask, який з URL повинен запускати нашу функцію, тобто налаштовуємо маршрут. Для запуску локального сервера з нашим застосунком, використовуємо функцію app.run(). Завдяки конструкції if __name__ == '__main__'.

HTTP (протокол, на якому спілкуються веб-застосунки) може використовувати різні методи для доступу до URL-адрес. За замовчуванням, @app.route() відповідає лише на запити типу GET, але це можна змінити, забезпечивши декоратор @app.route() аргументом methods. HTTP-метод повідомляє серверу, що хоче зробити клієнт з запитуваної сторінкою. В нашому випадку використовуватимемо два методи:

– GET – буде повертати головну сторінку index.html, де реалізовані за допомогою html та css: головне меню, де міститься опис та посилання на список всіх фільмів; сторінка має назву системи, поля для вводу фільмів та оцінку, кнопку «Submit» та тематичний дизайн.

– POST – викликатиме функції рекомендацій, які реалізовані в модулі contentbased_recom.py та collaborative_recom.py, якщо назви коректні, то повертатиме сторінку positive.html, в якому реалізовано меню, таке ж саме, як і на головній сторінці; назву системи, списки рекомендацій та тематичний дизайн. Якщо назви неправильні, то повертатиме сторінку negative.html, де – повідомлення про помилку та можливість повернення на головну сторінку для повторного введення.

Для запуску всієї системи вводимо вбудовану консоль VS Code python -m flask run, тоді запуститься вбудований локальний сервер, переходимо по посиланню http://127.0.0.1:5000/ та відкриється головна сторінка рекомендаційної системи.

Інтерфейс реалізованої системи рекомендацій та інструкцію користувача можна переглянути в Додатку В.

4.5. Висновки до розділу 4

Реалізовано всі модулі, які були спроєктовані в попередньому розділі. Бекенд, що відповідає за здійснення функціонування внутрішньої частини веб-застосунку, а саме: реалізація двох моделей рекомендацій, успішно працюють. Клієнтська сторона призначеного для користувача інтерфейсу, виконана вдало, тому зовнішнє представлення і внутрішня реалізація не конфліктують, що дає змогу проаналізувати отримані результати у наступних розділах.

РОЗДІЛ 5
АНАЛІЗ РЕЗУЛЬТАТІВ ТА ЇХ ПОРІВНЯННЯ

5.1. Результати та порівняння

При генерації рекомендації для введеного фільму Toy Story та при оцінці 3 для user-based і item-based рекомендаціях, отримали три списки, кожен із яких складається з 10 схожих фільмів, що можна побачити в табл. 5.1.

Таблиця 5.1

Список рекомендацій фільмів

№	Content-based	Item-based	User-based
1.	Toy Story 3	Toy Story 2 (1999)	Toy Story 2 (1999)
2.	Song of the Sea	Groundhog Day (1993)	Jurassic Park (1993)
3.	Inside Out	Independence Day (a.k.a. ID4) (1996)	Independence Day (a.k.a. ID4) (1996)
4.	Finding Nemo	Willy Wonka & the Chocolate Factory (1971)	Star Wars: Episode IV - A New Hope (1977)
5.	Aladdin	Mission: Impossible (1996)	Forrest Gump (1994)
6.	Monsters, Inc.	Nutty Professor, The (1996)	Lion King, The (1994)
7.	Up	Bug's Life, A (1998)	Star Wars: Episode VI-Return of the Jedi (1983)
8.	Zootopia	Lion King, The (1994)	Mission: Impossible (1996)
9.	The Grand Budapest Hotel	Babe (1995)	Groundhog Day (1993)
10.	Mary and Max	Monsters, Inc. (2001)	Back to the Future (1985)

Якщо пробігтись очима по таблиці, то відразу можна побачити, що в трьох списках наявні декілька схожих фільмів, а саме: Monsters, Inc., Toy Story 2,

Independence Day (a.k.a. ID4) (1996), Mission: Impossible (1996), Lion King, The (1994). В content-based міститься 1 фільм і 9 мультфільмів, item-based – 5 фільмів і 5 мультфільмів, user-based – 8 фільмів та 2 мультфільмів.

Виникає питання, як зрозуміти, чи релевантні рекомендації. Важливим кроком у будь-якої моделі машинного навчання є оцінка точності моделі. Для оцінки якості роботи існує безліч метрик якості. В основному, це метрики оцінки точності передбачуваного значення і реального. Розглянемо дві найпопулярніші метрики для оцінки методів рекомендаційних систем – MAE і RMSE.

RMSE (Root Mean Squared Error, середня квадратична помилка) – помилка обчислюється як корінь з суми квадратів різниць між передбачуваним значенням і реальним значенням, що розраховується:

$$RMSE = \sqrt{\frac{\sum_{i \in n}(P_i - R_i)^2}{n}}, \quad (5.1)$$

де n – це кількість елементів, P – передбачуване значенням, R – реальне значення. Тобто для кожного елемента розраховується квадрат відхилень (різниця між фактом і прогнозом, зведена у квадрат). Потім розраховується середнє арифметичне (сума квадратів відхилень, поділена на кількість) і витягуємо корінь з отриманого результату.

MAE (Mean Absolute Error, середня абсолютна помилка) – помилка оцінюється як різниця між прогнозом і реальною оцінкою по модулю:

$$MAE = \frac{\sum_{i \in n}|P_i - R_i|}{n}, \quad (5.2)$$

де n – це кількість елементів, P – передбачуване значенням, R – реальне значення. Вона означає, що всі індивідуальні відмінності зважені однаково в середньому. Як підсумок, MAE не така чутлива до викидів, як RMSE. Тому в нашому випадку будемо використовувати саме середню квадратичну помилку.

Отже, при введенні фільму Toy Story та при оцінці 3, ми отримали не лише список фільмів, а й міру схожості кожного виведеного фільму, для того, щоби порахувати RMSE. Нашими передбаченими значеннями будуть десять одиниць, адже як кореляція Пірсона, так і косинусна подібність прагне до 1, що означає дуже

високий позитивний зв'язок між двома елементами, а тому P = (1, 1, 1,1,1, 1, 1, 1, 1, 1), а R_{CB} = (0.228217, 0.17541, 0.166090, 0.146385, 0.143019, 0.136930, 0.136930, 0.136930, 0.129099, 0.122474) для content-based; R_{IB} = (0.230881, 0.180770, 0.179237, 0.178657, 0.176423, 0.175147, 0.172716, 0.172124, 0.170568, 0.165407) для item-based; та R_{UB} = (0.286301, 0.282818, 0.282131, 0.278694, 0.273548, 0.270573, 0.270545, 0.269456, 0.267084, 0.265191) для user-based. Розраховуємо кожне значення RMSE, підставляючи відповідні P та R. Результати обчислень містяться в таблиці 5.2.

Таблиця 5.2

Результати обчислень метрики якості RMSE

Метод рекомендацій	RMSE
Content-based	0.84837
Item-based	0.81999
User-based	0.72540

Отож, найкращим на наших даних виявився user-based метод, на другому місці item-based та на останньому content-based. А тому і косинусна відстань виявилась дієвою, аніж кореляція Пірсона. Розглянемо в чому проблема кожного методу.

Зважаючи на аналіз результатів, наша побудована content-based модель, виявилися емпірично менш точною при виробленні рекомендацій, ніж колаборативна, тобто зниження точності. Оскільки функціональне представлення фільмів певною мірою розробляється вручну, цей метод вимагає великого знання предметної області, що також залежить від повноти обраного набору даних. Отож модель може бути настільки хороша, наскільки гарні її характеристики, створені вручну. Тобто не можемо виключити можливість того, що фільми, які не досліджували, не більше актуальні, ніж ті, які вивчили.

Є величезні плюси – не вимагає ніяких даних про інших користувачів, бо рекомендації специфічні для цього користувача, це спрощує масштабування для великої кількості користувачів. А також, показує конкретні інтереси користувача і рекомендує нішеві елементи, які цікаві небагатьом іншим. На відміну від спільної

фільтрації, якщо у фільмів досить опису, уникаємо проблеми «холодного старту», тобто можна відразу надати рекомендацію.

Створена колаборативна фільтрація має декілька основних проблем. Розрідженість user-item матриці, адже, якщо подивитись попередні розділи, більшість користувачів не ставить оцінки фільму, у комірках матриці стоять нулі. Наприклад, тільки активні користувачі ставлять оцінки фільмам, у такий спосіб навіть найпопулярніші фільми можуть мати низьку кількість оцінок. Тобто рідкість даних впливає на якість рекомендацій користувача, а також посилює проблему холодного старту. «Холодний старт» – при додаванні нового фільму до даних, поки хтось його не оцінить, алгоритм не буде цей фільм рекомендувати. Також і з користувачем, поки не має інформації про його переваги, не можливо віднести до певної групи. Масштабування може стати проблемою в майбутньому для набору даних, що зростає, оскільки складність може стати занадто великою.

Плюси – метод незалежний від контенту, тому здатний точно рекомендувати фільми, не вимагаючи «розуміння» самих фільмів, у порівнянні з контентним методом, він більш точний, може дати більш релевантні рекомендації.

Отже, user-based алгоритм: холодний старт – нові об'єкти нікому не рекомендуються; нічого рекомендувати новим, або нетиповим користувачам, які нічого не оцінили. Item-based алгоритм: залишається брак у вигляді холодного старту і водночас рекомендації стають тривіальними, ресурсомісткість обчислень методу висока, тобто для прогнозів необхідно тримати в пам'яті оцінки всіх користувачів.

5.2. Пропозиції щодо поліпшення методів на основі результатів

Оскільки content-based система заснована на змісті, з огляду на повноту обраного набору можна поліпшити результати рекомендацій. Тобто, якщо опис фільму буде мати додаткові посилання на зовнішні джерела, то можна аналізувати також всю пов'язану з ним сторонню інформацію. Або при формуванні вектора

ключових слів замість окремого слова можна використовувати n-грами (послідовні пари слів, трійки тощо), наприклад, якщо у нас є фраза «подобається» і фраза «не подобається», у них протилежний зміст. Після того, як порахували всі потрібні нам n-грами, користуємося тими ж самими підходами, що й раніше, тільки буде не одне слово, а два, три тощо. Це зробить нашу модель більш деталізованою.

В алгоритмі рекомендацій спільної фільтрації ключовим моментом в нашому випадку є розрахунок подібності між користувачами user-based і фільмів item-based. Але при аналізуванні роботи методу виявилося, що переважна більшість оцінок невідома, а тому розрідженість матриці оцінок досить висока. З іншого боку, дані, що вже є в матриці досить суб'єктивні. Деякі користувачі – оптимісти, і їхні оцінки завжди високі, в інших користувачів оцінки завжди занижені. Для того, щоб оптимізувати продуктивність видачі рекомендацій, важливо нормалізувати оцінки.

В user-based – це простий і логічний підхід, наблизити новий рейтинг, як середній рейтинг даного користувача плюс відхилення від середніх рейтингів інших користувачів, зважених цими самими вагами, що розраховується за формулою:

$$\widehat{r_{ui}} = \overline{r_u} + \frac{\sum_{v \in K}(r_{vi} - \overline{r_v})sim(uv)}{\sum_{v \in K}|sim(uv)|}, \quad (5.3)$$

де $\widehat{r_{ui}}$ – рекомендації для користувача *u* для фільму *i*, $\overline{r_u}$ – середня оцінка, що виставив користувач *u*, *sim(uv)* – обрана міра схожості між користувачем *u* і *v*, *K* – множина найбільш схожих користувачів до *u*, r_{vi} – оцінка користувача *v* для фільму *i*, $\overline{r_v}$ – середня оцінка користувача *v*.

Для item-based рекомендацій усе абсолютно еквівалентно, рейтинг фільму *i* для користувача *u* можна передбачити, використовуючи просте середньозважене значення:

$$\widehat{r_{ui}} = \overline{r_i} + \frac{\sum_{v \in K}(r_{vj} - \overline{r_j})sim(ij)}{\sum_{v \in K}|sim(ij)|}, \quad (5.4)$$

де $\overline{r_i}$ – середня оцінка, що поставлена фільму *i*, а *sim (i j)* – міра схожості фільмів *i* та *j*.

5.3. Висновки до розділу 5

У розділі проведено аналіз результатів трьох методів рекомендацій. При перевірці релевантності отриманих списків фільмів, використовувалась метрика RMSE, яка продемонструвала, що найкращим на наших даних виявився user-based метод, а тому міра косинуса кута краще працює на наших даних. Після виявлення плюсів та мінусів кожного розробленого алгоритму, були надані рекомендації щодо їх поліпшення.

ВИСНОВКИ

У даній дипломній роботі проведено аналіз предметної галузі та її актуальності. У результаті проаналізовані основні алгоритми та підходи до прогнозування рекомендацій. Описані характеристики, переваги та недоліки кожного. Ознайомившись з такими компаніями, як Netflix та Amazon, стало зрозуміло, що проблема створення рекомендаційних систем є актуальною. Адже кожна велика компанія, яка використовує цю технологію, має похибки та стараються кожного разу поліпшити алгоритм виконання, тому як це залежить від попиту на товар та в підсумку на прибуток.

Є три основні підходи до побудови систем рекомендацій – це спільна фільтрація, фільтрація на основі вмісту та гібридна. Для того, щоб порівняти роботу декількох підходів вирішено обрати фільтрацію на основі вмісту та спільну (memory-based – item та user-based). Адже вони цілком підходять для вирішення поставленої мети – отримати 10 релевантних рекомендацій фільмів.

У ході огляду підходів, виявилось, що міра схожості між об’єктами відіграють одну з головний ролей. Двома популярнішими мірами є косинусна відстань та кореляція Пірсона, а також зважаючи на емпіричне дослідження, де йдеться, що ці дві міри дають кращі результати, обрано саме їх, аби також порівняти між собою.

Після огляду відповідної літератури та математичної моделі створено належний алгоритм заснований на content-based та collaborative підході. Відповідно до використання машинного навчання основними кроками для двох моделей є: вибір даних, тобто списків фільмів; попереднє опрацювання даних; скласти набір критеріїв; порівняти їх; дізнатися, який елемент подобається користувачеві; зіставити ці дані; отримати рекомендації.

Тобто проблем з цим етапом не виникало, адже по суті алгоритми є інтуїтивними. Обрано створювати веб-застосунок, тому подальшими кроками було визначення вимог, проєктування поведінки системи та архітектури за допомоги UML діаграм. Ця частина енерговитратна та вимагає відповідних знань. Завершальним

етапом стала практична реалізація веб-застосунку, попередньо обравши мову програмування та фреймворк. Виходячи з опрацьованої інформації, вибір пав на Python, адже широко використовується для машинного навчання, простий в розумінні та має велику кількість бібліотек для маніпулювання даними. Також для створення веб-сторінки обрано HTML та CSS.

Отже, у результаті роботи розроблено веб-застосунок для фільмів на основі content-based та collaborative підходів, клієнтська сторона отримала зручний та коректний інтерфейс, а бекенд – реалізовані алгоритми. Порівнюючи методи, аби зрозуміти, які із них дають найрелевантніші рекомендації, обрано метрика RMSE. Найкращу оцінку якості рекомендацій отримав user-based метод – 0.72540; item – 0.81999; content-based – 0.84837 відповідно. А тому міра, яка дає кращі результати на наших даних виявилась косинусна відстань.

Після порівняння надані рекомендації щодо поліпшення розроблених алгоритмів, для content-based – звернути увагу на повноту обраного набору або використання n-грами; для user та item-based – щоб оптимізувати продуктивність видачі рекомендацій, важливо нормалізувати оцінки. Алгоритм при цьому кардинально не змінюється, тобто: створюємо матрицю item-user; обчислюємо подібність між елементами; знайдемо k-найближчих сусідів за подібністю; прогнозуємо рейтинг, використовуючи формулу 5.3; виводимо рекомендації.

Найбільш подібними до створеної системи є IMDB та MEGOGO, але вони не використовують вміст фільмів, що є мінусом. Результати цієї роботи може бути використаним будь-яким сучасним веб-сервісом, що потребує ефективної рекомендаційної стратегії.

Загалом, поставлена мета була успішно виконана в процесі роботи. Однак надалі можливе проведення оптимізації та модифікації рекомендаційної системи, з метою підвищення точності результатів, доцільно звернути увагу на проблему «холодного старту». Виконаний аналіз показує, що не існує універсального алгоритму, потрібно з кожним разом минати поріг «суб’єктивності» рекомендацій, адже на поведінку людини впливає більше факторів, ніж подобається чи не подобається в конкретний момент часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to recommender systems - Towards Data Science. – [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

2. Sarwar B. Item-Based Collaborative Filtering Recommendation Algorithms./Sarwar B., Karypis G., Konstan J., and Riedl J.//Hong Kong WWW10 Conference: «WWW10», 2001, Hong Kong: materials. – Hong Kong: WWW10 – 2001.– 285-289 с.

3. Исследование метода коллаборативной фильтрации на основе сходства элементов: наукова стаття Е.Е. Пятикоп, канд.техн.наук, м. Маріуполь, 2014. – 109 с.

4. «Наука онлайн»./ Рекомендательные системы в дистанционном обучении //Александр Ерошенко – 2018. – 3-7 с.

5. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence – open access publishing for the scientific community. [Електронний ресурс]. – Режим доступу: https://www.hindawi.com/journals/aai/2009/421425/.

6. Міжнародний науковий журнал./Покращення результатів роботи рекомендаційних систем за допомогою алгоритму SVD//Мазурік Олексій Юрійович – 2015. – 62 с.

7. About Amazon - Amazon. – [Електронний ресурс]. – Режим доступу: https://www.aboutamazon.com//.

8. Как работает система рекомендаций Netflix - Netflix. – [Електронний ресурс]. – Режим доступу: https://help.netflix.com/ru/node/100639.

9. Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications, /Edited by Venkat N. Gudivada, C.R. Rao – 2018. – 331-401 с.

10. Vector space model - Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Vector_space_model.

11. ScienceDirect./ Empirical comparison of local structural similarity indices for collaborative-filtering-based recommender systems.//Qian-Ming Zhanga, Ming-Sheng Shanga, Wei Zenga, Yong Chena, Linyuan L. – 2010. – 1888-1895 с.

12. DeepAI. – [Електронний ресурс]. – Режим доступу: https://deepai.org/machine-learning-glossary-and-terms/cosine-similarity.

13. Pandas. – [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org/.

Додаток А
Алгоритми

А.1. Алгоритм для CBF

1. Зчитуємо файл IMDB_Top250.csv.
2. Попередня обробка даних.
   2.1 . Обираємо головні рядки – Назва, Жанр, Режисер, Актори, Опис сюжету.
   2.2 . Перетворення імен акторів і режисерів в окремі слова, щоб вони були унікальними, відкидаємо коми та отримуємо три імені.
   2.3 . Об’єднуємо імена та фамілії для кожного актора, щоб це було одне слово.
   2.4 . Витяг ключових слів з опису сюжету.
      2.4.1 Створюємо новий стовпець.
      2.4.2 У циклі проходимось по всім описам фільмів:
         – створення функції, яка знаходить ключові слова, відкидаючи стоп слова;
         – отримання та привласнення ключових слів новому стовпцю.
      2.4.3 Об’єднуємо всі стовпці Назва, Жанр, Режисер, Актори, Опис сюжету та Ключові слова в один стовпець (вектор).
3. Створення матриці підрахунку для вектору.
4. Створення матриці косинусної подібності.
5. Створення серії для назв фільмів, щоби вони відповідали індексам.
6. Створення функції рекомендації – приймає назву фільму як вхідні дані і повертає 10 кращих рекомендованих фільмів.
   6.1 . Оголошуємо масив рекомендацій.
   6.2 . Отримаємо індекс фільму, який відповідає назві фільму.
   6.3 . Створення серії з оцінками подібності в порядку убування.
   6.4 . Отримати індекси 10 більш схожих фільмів.
   6.5 . Заповнити список назвами 10 кращих відповідних фільмів.
   6.6 . Повернути список 10 кращих фільмів.
7. Вивід списку рекомендацій (10 фільмів схожих до введеного).

А.1. Алгоритм для CF

1. Зчитуємо файли ratings.csv та movies.csv.
2. Попередня обробка даних:
   - об’єднуємо таблиці та видаляємо стовпці genres та timestamp, в результаті таблиця зі стовпцями – movieId, title, userId, rating.
3. Створення user-item матрицю, в якій індекс userId, стовпець movieId та значення rating:
   - видалення фільмів, у яких є менш як 10 користувачів, які оцінили його, заповнення інших, що залишились NaN 0.
4. Для item-based:
   4.1 . Створення матриці подібності фільмів за допомоги кореляції Пірсона.
   4.2 . Зберігання матриці у .csv файлі.
   4.3 . Створення функції рекомендації, що приймає назву фільму, рейтинг як вхідні дані і виводить 10 схожих фільмів в порядку зменшення:
   - отримання фільму і оцінки;
   - віднімання від оцінки константу 2.5;
   - знаходження користувачів, яким сподобався введений фільм в матриці подібності, які фільми саме їм сподобались, витяг цих фільмів;
   - сортування списку фільмів та вивід лише 10 схожих в порядку зменшення.
5. Для user-based:
   1.1 Створення матриці подібності користувачів за допомоги косинус кута.
   1.2 Створення функції рекомендації, що приймає назву фільму, рейтинг як вхідні дані і виводить 10 схожих фільмів в порядку зменшення:
   - отримання фільму і оцінки;
   - віднімання від оцінки константу 2.5;
   - находження списку користувачів схожих за введеною оцінкою;
   - отримання усіх фільмів, що сподобались цьому списку користувачів;
   - сортування списку фільмів та вивід лише 10 схожих в порядку зменшення.
6. Вивід списку рекомендацій (10 схожих фільмів для item-based та 10 фільмів для user-based).

Додаток Б
Код програмних модулів

Б.1 Модуль рекомендацій для CBF

```
def data_processing():
    pd.set_option('display.max_columns', 100)
    df = pd.read_csv('D:\IMDB_Top250.csv')
    df = df[['Title', 'Genre', 'Director', 'Actors', 'Plot']]
    moviesList = list(df['Title'].values.flatten())
    df2 = pd.read_csv('D:\IMDB_Top250.csv')
    df2= df2[['Title', 'tomatoURL']]
    df2.set_index('Title', inplace = True)
    df['Actors'] = df['Actors'].map(lambda x: x.split(',')[:3])
    df['Genre'] = df['Genre'].map(lambda x: x.lower().split(','))
    df['Director'] = df['Director'].map(lambda x: x.split(' '))
    for index, row in df.iterrows():
        row['Actors'] = [x.lower().replace(' ', '') for x in row['Actors']]
        row['Director'] = ''.join(row['Director']).lower()
    df['Key_words'] = ""
    for index, row in df.iterrows():
        plot = row['Plot']
        r = Rake()
        r.extract_keywords_from_text(plot)
            key_words_dict_scores = r.get_word_degrees()
        row['Key_words'] = list(key_words_dict_scores.keys())
    df.drop(columns=['Plot'], inplace = True)
    df.set_index('Title', inplace = True)
        df['bag_of_words'] = ''
    columns = df.columns
    for index, row in df.iterrows():
        words = ''
        for col in columns:
```

```
            if col != 'Director':
                words = words + ' '.join(row[col])+ ' '
            else:
                words = words + row[col]+ ' '
        row['bag_of_words'] = words
    df.drop(columns = [col for col in df.columns if col!= 'bag_of_words'], inplace = True)
    ndf = pd.merge(df, df2, on=('Title'), how ='left')
    return ndf, moviesList
def get_similarity():
    count = CountVectorizer()
    count_matrix = count.fit_transform(ndf['bag_of_words'])
    indices = pd.Series(ndf.index)
    indices[:5]
    cosine_sim = cosine_similarity(count_matrix, count_matrix)
    return indices, cosine_sim
indices, cosine_sim = get_similarity()
def recommendations(title, cosine_sim = cosine_sim):
    recommended_movies = []
    url = []
    idx = indices[indices == title].index[0]
     score_series = pd.Series(cosine_sim[idx]).sort_values(ascending = False)
    top_10_indexes = list(score_series.iloc[1:11].index)
    rop_10_sim = list(score_series.iloc[1:11])
    for i in top_10_indexes:
        recommended_movies.append(list(ndf.index)[i])
    for i in top_10_indexes:
        url.append(list(ndf['tomatoURL'])[i])
    recom = pd.DataFrame(columns=['Title','tomatoURL'])
    recom['Title'] = recommended_movies
    recom['tomatoURL'] = url
    return recom
```

Б.1 Модуль рекомендацій для CF

```
def data_processing():
    ratings = pd.read_csv('ratings.csv')
    movies = pd.read_csv('movies.csv')
    moviesList2 = list(movies['title'].values.flatten())
   ratings = pd.merge(movies,ratings).drop(['genres','timestamp'],axis=1)
    n_users = ratings['userId'].nunique()
    n_items = ratings['movieId'].nunique()
    user_ratings = ratings.pivot_table(index=['userId'],columns=['title'],values='rating')
    user_ratings = user_ratings.dropna(thresh=10,axis=1).fillna(0)
    return user_ratings, moviesList2
def get_similar_movies(movie_name,user_rating):
    user_ratings, moviesList2 = data_processing()
    item_simil_df = user_ratings.corr(method='pearson')
    item_similarity_df = item_simil_df.where(item_simil_df.values !=
    np.diag(item_simil_df),0,item_simil_df.where(item_simil_df.values !=
    np.flipud(item_simil_df).diagonal(0),0,inplace=True))
    item_similarity_df.to_csv('item_similarity_df.csv')
    similar_score = item_similarity_df[movie_name]*(float(user_rating)-2.5)
    similar_movies = similar_score.sort_values(ascending=False)
    return similar_movies
def getRecomForItems(movie, rating):
    try:
        similar_movies = pd.DataFrame()
        similar_movies =
        similar_movies.append(get_similar_movies(movie,rating),ignore_index=True)
        all_recommend = similar_movies.sum().sort_values(ascending=False)
        m = all_recommend[0:10].to_string()
        m = m.split("\n")
        all_recom=[]
        for i in m:
            i = i.split("  ")
            all_recom.append(i[0])
```

```
            return all_recom
        except:
            return("No recommendation found:(")
    def get_similar_users():
        user_ratings, moviesList2 = data_processing()
        user_similarity = cosine_similarity(user_ratings.T)
        np.fill_diagonal(user_similarity, 0)
        user_similarity_df                                                    =
pd.DataFrame(user_similarity,index=user_ratings.columns,columns=user_ratings.columns)
        return user_similarity_df
    def getRecomForUsers(movie_name,ratings):
        try:
            user_similarity_df = get_similar_users()
            similar_score = user_similarity_df[movie_name]*(ratings-2.5)
            similar_score = similar_score.sort_values(ascending=False)
            sim = similar_score[0:10].to_string()
            sim = sim.split("\n")
            all_recom=[]
            for i in sim:
                i = i.split("  ")
                all_recom.append(i[0])
            return all_recom
        except:
            return("No recommendation found:(")
```

Додаток В
Інструкція користувача

1. Вимоги до програмного забезпечення:
   - Операційна система сімейства Windows (не нижче Windows 7)/Linux/Mac OS.
   - Підключення до мережі інтернет.
   - Веб-браузер Mozilla Firefox (не нижче 8.х версії).
2. Перший запуск програми.
   2.1 Заходимо в папку проєкту, знаходимо файл app з розширенням ру.
   2.2 Запускаємо файл.
   2.3 У консолі з'явиться посилання http://127.0.0.1:5000/, перейти по ньому та за допомогою веб-браузера виконається підключення.
3. Ознайомлення з системою програми.

Буде виконано перехід до «Головної сторінки», з якої надається доступ до системи з наступним переліком (див. рис. Д.1):

Рис. Д.1. «Головна сторінка»

- назви системи;
- поля для вводу фільмів та оцінки другого;
- кнопки «Submit»;
- кнопки «Меню».

4. Початок роботи.
   4.1 Отримання рекомендацій:
   - Вводимо, в обов'язкові поля (див. рис. Д. 2), назву фільмів та оцінку другого (див. рис. Д. 3).

Рис. Д.2. Обов’язкові поля

Рис. Д.3. Введення назви фільму

– Натискаємо кнопку «Submit».
– Якщо фільми введені коректно, користувач переходить на сторінку «Рекомендацій» та отримує список 10 фільмів для content-based, 10 item-based, 10 user-based (див рис. Д.5), також є можливість перейти по посиланню на сайт фільму (див. рис. Д. 4).

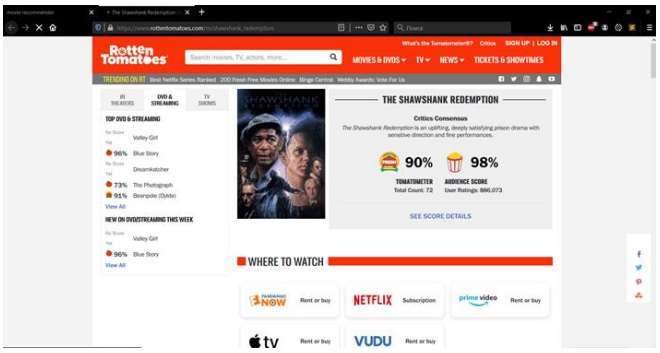


Рис. Д.4. Перехід на сайт фільму

– Якщо фільм некоректно введений користувач переходить на сторінку «Помилки» (див. рис. Д. 6).

Сторінка «Рекомендацій» складається з:

Рис. Д.5. Сторінка «Рекомендацій»

– назви recommendations;
– списку рекомендацій фільмів, у сумі 30;
– кнопки «Меню».

Сторінка «Помилки» складається з:

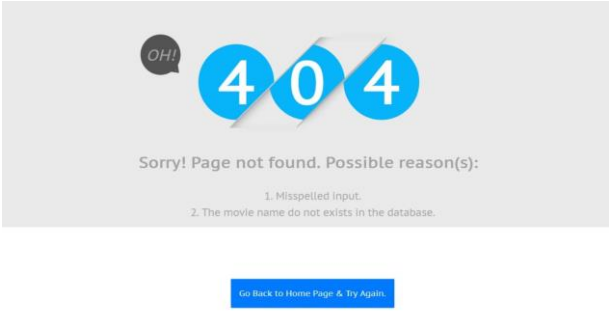


Рис. Д.6. Сторінка «Помилки»

– номер помилки ;
– причини помилки;
– кнопки «Go Back to Home Page & Try Again», при натисненні кнопки повертає на головну сторінку.

Перегляд меню системи:

Для перегляду «Меню» необхідно натиснути кнопку меню у верхньому правому куті. Меню матиме вигляд:

Рис. Д.7. «Меню» системи

- інформацію про систему;
- посилання на список фільмів, яка використовує система;
- посилання на сторінку розробника та контакти;
- кнопки «Меню».

Додаток Д
Software Architecture Document

Recommendation system based on content and collaborative filtering. Software Architecture Document (SAD)

CONTENT OWNER: Lazukina Polina

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
• 1.0.0	• 1.0.0	• 01.06.2021

Table of Contents

1 Documentation Roadmap

1.1 Document Management and Configuration Control Information

- Revision Number: 1.0.0
- Revision Release Date: 01.06.2021
- Purpose of Revision: first release of the document

1.2 Purpose and Scope of the SAD

This SAD specifies the software architecture for the recommendation system. All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents. The Software Architecture Document (SAD) provides a comprehensive architectural overview of the recommendation system. It presents a number of different architectural views to depict the different aspects of the system. This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. These structures will be represented in the views of the software architecture that are provided in Section **Ошибка! Источник ссылки не найден.**.

1.3 Viewpoint Definitions

As required by ANSI/IEEE 1471-2000, this SAD employs a stakeholder-focused, multiple view approach to architecture documentation. A viewpoint identifies the set of concerns to be addressed and a view is a viewpoint applied to a system. In current section are presented and defined viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns. In the next subsections, each viewpoint is shortly presented.

Table 1: Stakeholders and Relevant Viewpoints

Stakeholder	**Viewpoint(s) that apply to that class of stakeholder's concerns**
Developer	Module decomposition, Data flow, Deployment
Maintainer	Module decomposition, Data flow, Deployment
End user	Data flow, Deployment
Acquirer	Module decomposition, Deployment

1.3.1 Module decomposition viewpoint definition

1.3.1.1 Abstract. Views conforming to the module decomposition viewpoint partition the system into a unique non-overlapping set of hierarchically decomposable implementation units – modules.

1.3.1.2 Stakeholders and Their Concerns Addressed. Stakeholders and their concerns addressed by this viewpoint include

- developers, who use the elements to produce a running version of the system;
- maintainers, who are tasked with modifying the software elements;
- acquirers, who are concerned that projected changes to the system over its lifetime can be made economically by confining the effects of each change to a small number of elements.

1.3.1.3 Elements, Relations, Properties, and Constraints. Elements of the module decomposition viewpoint are modules, which are units of implementation that provide defined functionality. Modules are hierarchically decomposable; hence, the relation is "is-part-of." Properties of elements include their names, the functionality assigned to them and their software-to-software interfaces.

1.3.1.4 Language(s) to Model/Represent Conforming Views. UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

1.3.2 Data flow viewpoint definition

1.3.2.1 Abstract. Views conforming to the data flow (pipeline) viewpoint represent the system as a computational model in which components act as data transformers and connectors transmit data from the outputs of one component to the inputs of another.

1.3.2.2 Stakeholders and Their Concerns Addressed. Stakeholders and their concerns addressed by this viewpoint include

- developers, who use the elements to produce a running version of the system;
- maintainers, who are tasked with modifying the software elements;
- users, who will use the final system and need to know basic architecture overview.

1.3.2.3 Elements, Relations, Properties, and Constraints. Elements of the pipeline viewpoint are pipe and filter. Filter is a component that transforms data read on its input ports to data written on its output ports. Pipe is a connector that conveys data from a filter's output ports to another filter's input ports. The "attachment" relation associates filter output ports with data-in roles of a pipe, and filter input ports with data-out roles of pipes. Properties of elements include their names, the functionality assigned to them and the data transferred between them. The main constraint is that pipes connect filter output ports to filter input ports.

1.3.2.4 Language(s) to Model/Represent Conforming Views. (a) UML, activity diagram or (b) dataflow diagram.

1.3.3 Deployment viewpoint definition

1.3.3.1 Abstract. Deployment viewpoints allocate software elements, which are native to a component & connector style to the hardware of the computing platform on which the software executes.

1.3.3.2 Stakeholders and Their Concerns Addressed. Stakeholders and their concerns addressed by this viewpoint include

- developers, who use the elements to produce a running version of the system;
- maintainers, who are tasked with modifying the software elements;
- users, who will use the final system and need to know basic architecture overview;
- acquirers, who are concerned that projected changes to the system over its lifetime can be made economically by confining the effects of each change to a small number of elements.

1.3.3.3 Elements, Relations, Properties, and Constraints. Elements of the deployment viewpoint are software elements and environmental elements. Software element is an elements from a component & connector view. Environmental element is a hardware of the computing platform — processor, memory, disk, network and so on. Possible relations are “allocated-to”, “migrates-to”, “copy-migrates-to”, “execution-migrates-to”. Properties of elements include their names, the functionality assigned to them and the data transferred between them. The allocation topology is unrestricted. However, the required properties of the software must be satisfied by the provided properties of the hardware.

1.3.3.4 Language(s) to Model/Represent Conforming Views. (a) informal graphical notations that use boxes, circles, lines, arrows, and so on to represent the software and environmental elements; (b) UML, a deployment diagram is a graph of nodes connected by communication associations; (c) Architecture Analysis and Design Language (AADL) or (d) SysML which are architecture description languages that provide formal notations for describing deployment views.

2 Architecture Background

2.1 Problem Background

2.1.1 System Overview

The purpose of this system is to offer the user the different recommendations of films that he can choose from a variety of methods, which give recommendations films that he likes. The recommendation system also has a graphical user interface where he can enter two films and score the second, browse the system menu, and get three variants of film recommendations, each with the best ten matches.

The recommendation system has a graphical user interface. Where there are three web pages. "Home page", where the name of the system; a field for entering two movie titles and evaluating the second; "Submit" button; a menu with information about the system and a link to a list of all movies. "Error page", where: error information; return to the "main page". "Recommendation page", where: system name; top 10 movie recommendations as a list for content-based; top 10 movie recommendations in the form of a list for collaborative filtering (item and user based); menu with system information and a link to a list of all movies

It can be concluded that the user interacts with the system step by step and sequentially. If the user chooses to enter the first film, the second film and its rating, clicks on the "Submit" button, he goes to the recommendations page, if the user entered movies and rating are incorrect (incorrect name, entered with errors, not in the list, rating not within 1 -5), the user will go to the error page, where you can go back, namely to the main page.

If the user wants to get information about the system, he goes to the menu, which contains a description of the system and a link to a list of two data sets of movies used by the system, otherwise close the menu and return to the main page, or completely log out.

Software requirements: Windows family operating system (not less than Windows 7)/Linux/Mac OS; internet connection; Mozilla Firefox web browser (version 8.x).

Therefore, the system should be compact. Cost of the final system (including software and hardware) must be cheap enough (~60$). System should be modular and flexible to allow further expansion and modification, support software component changes.

2.1.2 Goals and Context

The web application will have a client-server architecture. As described in previous section, solution should be modular to support extension of the system and components change. For solution to be modular software architecture should be modular too. From description several modules can be formed:

- the Main module app.py, which will generate a web application in which Flask is imported and an instance of the Flask class is created, which will be our WSGI application;
- the Recommendation folder, where 2 recommendation modules, namely: contentbased_recom.py, in which 3 libraries are imported, data reading function, data conversion, pre-processing, get_similarity() function, where Count Vectorizer and Cosine Similarity function, as well as recommendations function; collaborative_recom.py, where 3 libraries are imported, data_processing() function, which performs pre-processing and calculation of similarity matrix,

get_similar_movies() - similarity calculation for movies and get_similar_users () - for users, two functions of recommendations getRecomForItems(), getRecomForUsers();

- the Model folder, where our movie data will be in .csv format;
- the View folder - 3 web pages, which consist of page structure (html) and appearance (css); index.html - will have an input field, "Submit" button, menu button, system description; positive.html - menu button, system description, list of recommendations; negative.html - information about the problem and the back button.

These modules can be changed in future. Since it will be a web application, it uses a client-server architecture that runs on a web server page, the WSGI configuration is built into the framework, and uses a web browser (client), interacting with the HTTP protocol. This choice is because it is easy to use, resistant to attacks, and has a quick response to user actions.

2.1.3 Significant Driving Requirements

The main key requirements are the speed of the solution and architecture modularity. The former can be achieved with efficient algorithm recommendations. The latter can be achieved with the modular style of system designing, and with the required level of control of cohesion and coupling between software functions.

Functional requirements, "what" the system must do:

1) introduction of the first film for content-based recommendation, introduction of the second film and its evaluation for collaborative filtering (films may coincide);

2) checking the films for correctness of input;

3) output error message;

4) output of three different film recommendations;

5) provide information on what kind of system it is and what data set is used;

Non-functional requirements, "what" should be the system:

1) the program must have a user interface:
    a. reliable performance (without errors);
    b. availability of the interface;
    c. efficiency;
    d. a transparent and expressive web interface in English, which must have:
        - "Main page", where it should be: the name of the system; a field for entering two movie titles and evaluating the second; "Submit" button; menu where information about the system and a link to the list of all movies will be placed;
        - "Error page", where: error information; return to the "main page";
        - "Recommendations page", where: system name; Top 10 movie recommendations as a list for content-based; Top 10 movie recommendations in the form of a list for collaborative filtering (item and user based); menu, where information about the system and links to a list of all movies;
2) entering movie titles in English;
3) ease of operation.

2.2 Solution Background

2.2.1 Architectural Approaches

Modularity of architecture can be achieved with help of software modules. Each module performs one task and can be extended or modified to meet new changes that can be presented with new requirements, hardware and/or software changes. Each module has open external inter-face seen to other modules to reduce coupling between modules and increase cohesion inside modules.

Primarily the division of responsibilities between the client and the server determines the model of client-server interaction. It is logical to separate three levels of operations:

- a level of data representation, which is essentially a user interface and is responsible for presenting data to the user and entering control commands from him;
- application level, which implements the basic logic of the application and at which the necessary information processing;
- a level of data management that provides data storage and access.

2.2.2 Analysis Results

Implemented all the modules that were designed in the previous section. The backend that is responsible for running the internal part of the web application, namely the implementation of the two models of recommendations, is working well. The client side of the user interface is successful, so the external representation and internal implementation do not conflict. Recommendations are given within 4 seconds, which is a good result.

To measure modularity metrics to measure cohesion and coupling were used. For coupling metrics, CE (efferent coupling) and CA (afferent coupling) were used. For cohesion, metric LCOM was used. As solution is small enough and has small amount of methods metrics show small values, which are interpreted as good results.

2.2.3 Requirements Coverage

Requirements addressed by the software architecture are processing speed of the system and system modularity. Processing speed is achieved with efficient algorithm. System modularity is addressed in flexibility, open for extension, possible component changes requirements and this aspect is achieved with module-based design of software. The user interface includes compliance with all requirements and the speed of providing recommendations and using the system.

The main requirements are covered and results of analysis provided in previous section.

3 Views

This section contains the views of the software architecture described in Section 1.3.

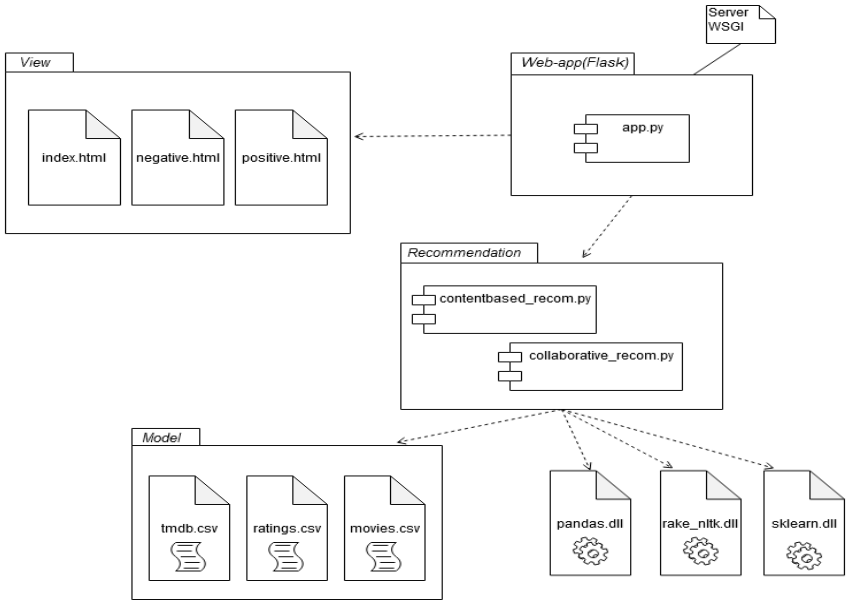
3.1 Module decomposition View

3.1.1 View Description

This view partitions the system into a unique non-overlapping set of hierarchically decomposable modules. The modules that described earlier are given on a diagram.

3.1. View Overview

View represented as diagram built based on UML package diagram notation.



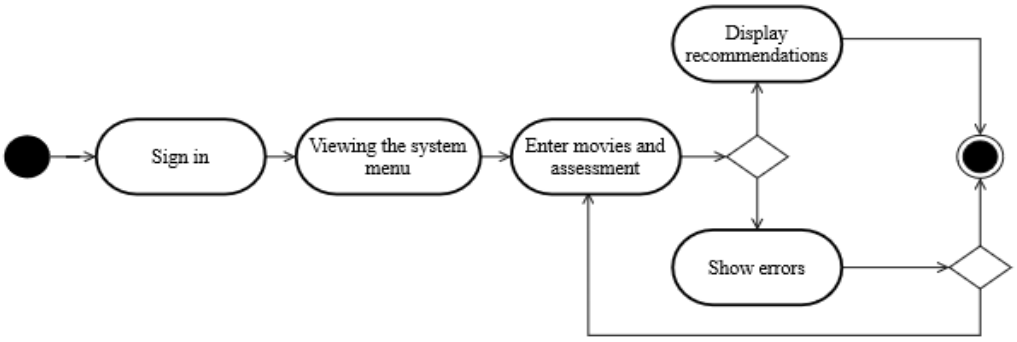
3.2 Data Flow View

3.2.1 View Description

This view represents the system as a computational model in which components act as data transformers and connectors transmit data from the outputs of one component to the inputs of another. System in given view presented as pipeline of modules described earlier.

3.2.2 View Overview

View represented as diagram built based on UML activity diagram notation.



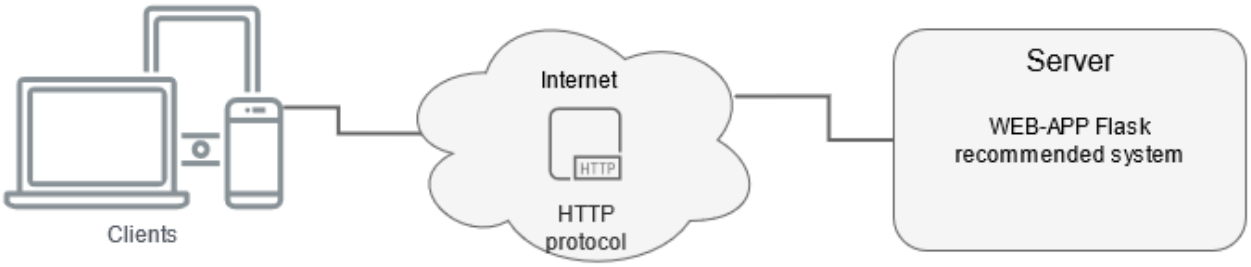
3.3 Deployment View

3.3.1 View Description

This view allocates software elements, which are native to a client-server architecture.

3.3.2 View Overview

View represented as diagram built with informal graphical notations that use boxes, circles, lines, arrows, and so on to represent the software and environmental elements for client-server architecture.



4 Referenced Materials

Clements 2010	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2010.
IEEE 1471	ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000.

5 Directory

5.1 Glossary

Term	Definition
software architecture	The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.
view	A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.
viewpoint	A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view.

5.2 Acronym List

API	Application Programming Interface; Application Program Interface; Application Programmer Interface
CF	Collaborative filtering
CBF	Content-based filtering
NLP	Natural Language Processing
IEEE	Institute of Electrical and Electronics Engineers
GUI	Graphical user interface
SAD	Software Architecture Document
UML	Unified Modeling Language