НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.912

«До захисту допущено»
Завідувач кафедри
_____Євгенія СУЛЕМА
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Інженерія програмного забезпечення мультимедійних
та інформаційно-пошукових систем»
зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Спосіб та програмне забезпечння для персонального
веб-сайту з використанням машинного навчання»

Виконала:
студентка ІІ курсу, групи КП-11мп
Фалілеєва Дар'я Миколаївна _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н,
Хіцко Яна Володимирівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПіСКС, к.т.н., доцент,
Боярінова Юлія Євгеніївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студентка _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Фалілеєвій Дар'ї Миколаївні

1. Тема дисертації «Спосіб та програмне забезпечення для створення персонального веб-сайту з використанням алгоритмів машинного навчання», науковий керівник дисертації Хіцко Яна Володимирівна, старший викладач кафедри ПЗКС, к.т.н., затверджені наказом по університету від «25» листопада 2022 р. № 4317−С

2. Термін подання студентом дисертації «14» грудня 2022 р.

3. Об'єкт дослідження: процес побудови рекомендаційних систем із використанням методів машинного навчання.

4. Предмет дослідження: моделі, методи та алгоритми створення рекомендаційних систем.

5. Перелік завдань, які потрібно розробити:
   - Проаналізувати існуючі методи та підходи до вирішення задачі персоналізації контенту;
   - провести порівняльний аналізтметодів та алгоритмів рекомендаційних систем;
   - формалізувати задачу формування рекомендацій на основі гібридного методу;
   - розробити ефективну модель рекомендації за допомогою гібридного методу;
   - розробити програмне забезпечення;
   - експериментально дослідити ефективність отриманої моделі

6. Перелік графічного (ілюстративного) матеріалу:
   - Схема бази даних;

- ER – модель бази даних;
- діаграма прецентентнів;
- діаграма діяльності;
- Схема архітектури нейронної мережі.

7. Перелік публікацій:
   - Тези доповіді “Спосіб та програмне забезпечення для створення персонального веб-сайту з використанням алгоритмів машинного навчання”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М. В., доцент кафедри ПЗКС к.т.н., доцент		

9. Дата видачі завдання «04» жовтня 2021 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.10.2021	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2021	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2022	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2022	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2022	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2022	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК−2022	05.11.2022	
8.	Оформлення текстової і графічної частини магістерської дисертації	04.12.2022	

Студент Дар’я ФАЛІЛЕЄВА

Науковий керівник Яна ХІЦКО

РЕФЕРАТ

Актуальність теми.

Компанії, які використовують рекомендаційні системи, можуть збільшити продажі за допомогою дуже персоналізованих пропозицій і підвищення якості обслуговування клієнтів. Більше того, компанії можуть залучати та утримувати клієнтів, розсилаючи електронні листи з посиланнями на нові пропозиції, що відповідають інтересам одержувачів, або пропозиціями фільмів та телешоу, що відповідають їхньому профілю.

Рекомендації зазвичай прискорюють пошук і полегшують користувачам доступ до контенту, що їх цікавить, а також дивують їх пропозиціями, які вони ніколи б не шукали. Користувач почне почуватися важливим і зрозумілим і з більшою ймовірністю придбає додаткові продукти або буде доваше залишатись на платформі. Знаючи, чого хоче користувач, компанія отримує конкурентну перевагу, а загроза втрати клієнта на користь конкурента знижується.

Об'єктом дослідження процес побудови рекомендаційних систем із використанням методів машинного навчання.

Предметом дослідження є моделі, методи та алгоритми створення рекомендаційних систем.

Метою дослідження є реалізація веб-додатку для персоналізації контенту користувачів з використанням запропонованого гібридного алгоритму.

Для її досягнення були виділені наступні задачі:

- Проаналізувати існуючі методи та підходи до вирішення задачі персоналізації контенту;
- провести порівняльний аналіз різних методів та алгоритмів рекомендаційних систем;

- формалізувати задачу формування користувацьких рекомендацій на основі гібридного методу;
- розробити ефективну модель рекомендації контенту за допомогою гібридного методу;
- розробити модель машинного навчання;
- розробити програмне забезпечення;
- експериментально дослідити ефективність отриманої моделі.

Результати та їх наукова новизна полягає в тому що:

- Було досліджено проблему персоналізації контенту;
- був проведений порівняльний аналіз існуючих алгоритмів персоналізації контенту, визначено їх переваги та недоліки;
- було розглянуто актуальні підходи до побудови рекомендаційних систем і визначено найбільш ефективні серед них;
- було розроблено гібридний алгоритм для фільтрації данних при формуванні особистих рекомендацій користувачу;
- створена модель машинного навчання, яка була використана для побудови веб-додатку для персоналізації контенту.

Практичне значення отриманих в магістерській дисертації результатів визначається можливим розширенням на інші класи систем, а не тількі для рекомендації фільмів. Наприклад, в електронній комерції при створенні інтернет-магазинів, порталу з книгами, музикою, дошки оголошень, тощо.

Апробація роботи. Основні положення та результати роботи були представлені та обговорювались на XIV науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК–2022.

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, п’яти розділів, висновків та додатків.

У вступі надано загальну характеристику рекомендаційних си
сфери та переваги застосування.

У першому розділі було проаналізовано наявні сервіси, які використовують рекомендаційні системи. Також була описана актуальність, сформована задача, яка буде проаналізована під час роботи над дисертацією. Було вказано призначення розробки рекомендаційної системи, цілі та задачі, які повинні бути досягнуті під час роботи з системою рекомендацій контенту.

У другому розділі було обґрунтовано поняття рекомендаційних систем, які необхідні дані для її побудови, а також основні проблеми, які можуть виникнути під час її створення. Було визначено основні переваги та недоліки даних методів, а також проведено порівняльний аналіз між ними. Також було описано основні типи метрик, які необхідні для оцінки ефективності, а також їх основні переваги та недоліки.

У третьому розділі було сформовано постановку задачі, обрано програмних забезпечень для реалізації веб застосунку, було детально описано функціональні можливості та вимоги до веб сейту та технічного забезпечення. Було опсано вискорівненву архітектуру програмного забезпечення, нейронної мережі, а також моделі машинного навчання, яка буде його реалізовувати.

У четвертому розділі були розроблені, проаналізовані та обрані відповідні програмні засоби для реалізації функцій додатку, а саме мова програмування, фреймворки, бібліотеки для створення інтерфейсу користувача, серверна частина додатку та реалізація алгоритмів персоналізації. Програмне забезпечення було створено за допомогою вибраних засобів розробки. В результаті роботи отримано сучасний, функціональний веб-додаток, який відповідає заданим програмним вимогам. Окремо були проведені експериментальні дослідження з використанням розробленої моделі машинного навчання.

У п'ятому розділі запропоновано бізнес модель, що дозволяє представити на ринку повноцінний програмний продукт із використ представлених у роботі напрацювань.

У висновку проаналізовано отримані результати даної магістерської роботи.

Робота виконана на 143 аркушах, містить 6 додатків та посилання на список використаних літературних джерел з 51 найменувань. У роботі наведено 25 рисунків та 5 таблиць

Ключові слова: персоналізація, рекомендаційна система, алгоритми машинного навчання, веб-додаток, клієнт-серверна архітектура, Python, JavaScript, React.

ABSTRACT

Actuality of theme.

Companies that use recommender systems can increase sales through highly personalized offers and improved customer service. Moreover, companies can attract and retain customers by sending emails with links to new offers that match the recipient's interests, or suggestions for movies and TV shows that match their profile.

Recommendations usually speed up searches and make it easier for users to access content they're interested in, as well as surprise them with offers they would never have looked for. The user will begin to feel important and understood and will be more likely to purchase additional products or prefer to stay on the platform. Knowing what the user wants, the company gains a competitive advantage, and the threat of losing the customer to a competitor is reduced.

The object of research is the process of building recommendation systems using machine learning methods.

The subject of research are models, methods and algorithms for creating recommender systems.

The purpose of the research is to implement a web application for personalizing user content using the proposed hybrid algorithm.

To achieve it, the following tasks were selected

- Analyze existing methods and approaches to solving the problem of content personalization;
- conduct a comparative analysis of various methods and algorithms of recommender systems;
- formalize the task of forming user recommendations based on the hybrid method;
- develop an effective content recommendation model using a hybrid method;
- develop a machine learning model;

- develop software;
- experimentally investigate the effectiveness of the obtained model.

The results and their scientific novelty are that:

- The problem of content personalization was investigated;
- a comparative analysis of existing content personalization algorithms was conducted, their advantages and disadvantages were determined;
- current approaches to building recommendation systems were considered and the most effective among them were determined;
- a hybrid algorithm was developed for data filtering when forming personal recommendations to the user;
- created a machine learning model that was used to build a web application for content personalization.

The practical value of the results obtained in the master's thesis is determined by the possible extension to other classes of systems, and not only for recommending movies. For example, in e-commerce when creating online stores, a portal with books, music, bulletin boards, etc.

Approbation of work. The main provisions and results of the work were presented and discussed at the XIV Scientific Conference of Master's and Postgraduate Students "Applied Mathematics and Computing" PMK–2022.

Structure and scope of work. The master's thesis consists of an introduction, five chapters, conclusions and appendices.

The introduction provides a general description of recommender systems, areas and benefits of application.

The first chapter analyzed existing services that use recommender systems. The relevance was also described, the task was formed, which will be analyzed during the work on the dissertation. The purpose of developing a recommender system, the goals and objectives that should be achieved when working content recommender system were specified.

In the second chapter, the concept of recommender systems, the necessary data for its construction, as well as the main problems that may arise during its creation were substantiated. The main advantages and disadvantages of these methods were determined, and a comparative analysis was carried out between them. It also described the main types of metrics that are needed to evaluate performance, as well as their main advantages and disadvantages.

In the third chapter, the statement of the problem was formed, software was chosen for the implementation of the web application, the functionality and requirements for the web application and technical support were described in detail. The correct architecture of the software and neural network, as well as the machine learning model that will implement it, were described.

In the fourth chapter, appropriate software tools were developed, analyzed and selected to implement the functions of the application, namely, the programming language, frameworks, libraries for creating the user interface, the server part of the application, and the implementation of personalization algorithms. The software was created using selected development tools. As a result of the work, a modern, functional web application that meets the specified software requirements was obtained. Separately, experimental studies were conducted using the developed machine learning model.

In the fifth chapter, a business model is proposed, which allows to present a full-fledged software product on the market using the developments presented in the work.

The results of this master's thesis are analyzed in the conclusion.

The work is completed on 143 sheets, contains 6 appendices and links to the list of used literary sources from 51 titles. The work contains 25 figures and 5 tables.

Keywords: personalization, recommender system, machine
algorithms, web application, client-server architecture, Python, JavaScript, F

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

PGM − графічна модель, або імовірнісна графічна модель.

MSE − коренева середньоквадратична помилка.

RMSE − середньоквадратична помилка.

MAE – середня абсолютна помилка.

mAP (англ. mean average precision) – середня точність.

ROC (англ. receiver operating characteristic) – робоча характеристика приймача.

CF – метод колаборативної фільтрації.

nDCG (англ. normalized discounted cumulative gain) – нормалізований дисконтний сукупний виграш.

HTML (Hypertext Markup Language) – мова розмітки гіпертексту.

CSS *(англ. Cascading Style Sheets)* – каскадні таблиці стилів.

БД – база данних.

ВСТУП

В еру діджиталізації кількість доступної інформації в Інтернеті збільшується щосекунди, кількість сайтів з тисячами, а то й мільйонами продуктів з'являться щодня, разом з цим і складність систем та програмних продуктів для коректної обробки також підвищується. Відповідно це впливає на кінцевих користувачів – в таких умовах дуже складно знайти необхідну інформацію швидко, концентрація розсіюється і в кінцевому результаті людина йде ні з чим, адже не може впоратись з величезним обсягом інформації.

Якщо компанія працює над поліпшенням персоналіфікації контенту, це допомагає користувачам знаходити те, що вони шукають, а іноді й пропонувати те, що вони навіть не уявляли, що воно було необхідно. При цьому підприємства можуть більше дізнатися про унікальні вподобання та інтереси кожного користувача, оптимізуючи виробництво у режимі реального часу, створювати прогнози на довгострокову перспективу.

За допомогою персоналізованих рекомендацій компанії можуть допомогти користувачам легко знаходити релевантні продукти на основі схожості продуктів, тенденцій, інтересів та поведінки з кінцевою метою стимулювати продажі, додаткові продажі, перехресні продажі, збільшення розміру кошика та більш високу середню вартість замовлення.

Декілька вдалих прикладів використання персональних рекомендацій:

- Netflix: 2/3 переглянутих фільмів є рекомендованими.
- Новини Google: рекомендації генерують на 38% більше кліків.
- Amazon: 35% продажів від рекомендацій.
- YouTube: 70% переглянутих відео є рекомендованими.
- Alibaba: рекомендації підвищують коефіцієнт конверсії на 20% [1].

Зростання важливості Інтернету як середовища для електронних і

ділових операцій послужила рушійною силою для розвитку технології рекомендаційних систем. Важливим каталізатором у цьому питанні є легкість, з якою Інтернет дозволяє користувачам надавати відгук про те, що їм подобається чи не подобається.

Наприклад, розглянемо сценарій контент-провайдера наприклад Netflix. У таких випадках користувачі можуть легко надати відгук простим кліком миші. Типова методологія надання зворотного зв'язку – це рейтинги, в яких користувачі вибирають числові значення з певної системи оцінювання (наприклад, п'ятизіркової системи оцінювання), які визначають їхні симпатії та антипатії до різних предметів. Інші форми зворотного зв'язку не такі явні, але їх навіть легше зібрати в веб-центричну парадигму. Наприклад, просту дію користувача, який купує або переглядає товар можна розглядати як схвалення цього елемента.

Такі форми зворотного зв'язку є звичайними і використовуються онлайн-продавцями, такими як Amazon.com, збирання такого типу даних абсолютно невимушене з точки зору трудомісткості, що вимагається від клієнта. Основна ідея систем рекомендацій полягає у використанні цих різноманітних джерел даних для визначення інтересів клієнтів.

Рекомендаційні системи, в першу чергу, спрямовані на осіб, яким бракує особистого досвіду чи компетентності, щоб оцінити велику кількість альтернативних елементів, які може запропонувати веб-сайт. Яскравим прикладом є книга система рекомендацій, яка допомагає користувачам вибрати книгу для читання.

Оскільки рекомендації зазвичай персоналізовані, різні користувачі або групи користувачів отримують різноманітні пропозиції. Крім того, бувають і неперсоніфіковані рекомендації. Їх набагато простіше згенерувати, і вони зазвичай публікуються в журналах або газетах. Типовий приклад − десять найкращих добірок книг, ресторанів, тощо.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Системи рекомендацій, як випливає з назви, – це системи, які рекомендують щось своїм користувачам. Це інструменти, які пропонують пропозиції щодо предметів, наприклад, що купувати, які пісні слухати чи який фільм варто переглянути.

Для забезпечення роботи систем рекомендацій необхідно виконати такі кроки:

- Збір даних – цей крок передбачає збір даних від користувачів або різних джерел, щоб система могла персоналізувати досвід користувача на основі отриманих даних. Це можна зробити двома способами:
  - По-перше, явно. Ці дані надаються користувачем навмисно у вигляді оглядів фільмів, рейтингів тощо.
  - По-друге, імпліцитно. Ці дані збираються лише від користувачів, але не навмисно. Він збирається з інших доступних потоків даних, таких як історія пошуку, історія замовлень, кліки тощо.
- Зберігання даних: після того, як дані зібрані, ми повинні зберігати ці дані ефективно та впорядковано. Даних зазвичай дуже багато, тому ними потрібно правильно керувати. Чим більше даних, тим краще система рекомендацій. Тип зібраних даних визначає їх зберігання. Це може бути стандартна база даних SQL або база даних без SQL.
- Фільтрація даних – після збереження даних наступним кроком є фільтрація даних. Нам потрібно відфільтрувати дані, щоб отримати відповідну інформацію, яка допоможе зробити остаточну рекомендацію. Дані фільтруються за допомогою одного з кількох існуючих алгоритмів [2].

1.1. Огляд існуючих рекомендаційних систем та їх підходів

1.1.1. Рекомендаційна система Netflix

Система Netflix показує персоналізовані рекомендації на основі кількох факторів, зокрема:

- Попередня взаємодія кожного користувача (наприклад, історія переглядів, пошуки та оцінки).
- Вибір інших учасників (особливо зі схожими смаками та уподобаннями).
- Інформація про конкретну назву (жанр, категорія, рік випуску тощо);
- Пристрій для перегляду відео на Netflix.
- Час перегляду [3].

У сукупності всі ці дані є корисною вхідною інформацією для алгоритмів рекомендацій Netflix. Вони обробляють і аналізують усі ці дані та використовують машинне навчання, щоб перетворити їх у корисні та точні рекомендації щодо фільмів.

Коли користувач створює новий обліковий запис Netflix, він повинен вибрати кілька назв фільмів, які йому подобаються. Це відправна точка для початку роботи алгоритму рекомендацій. Крім того, система рекомендацій, яку використовує Netflix, зосереджується на останніх виборах кожного користувача. Тож якщо кілька років тому користовача цікавили фантастичні фільми, а зараз він дивиться переважно романтичні фільми, алгоритм пропонуватиме переважно такі назви.

Система рекомендацій Netflix насправді дуже складна, і вона використовує різні технології та моделі машинного навчання, щоб надавати мільйонам користувачів точні пропозиції. Існує кілька алгоритмічних підходів, які включають:

- Навчання з підкріпленням (алгоритмам не потрібна попередня інформація; вони вивчають дані під час процесу).

- Нейронні мережі.
- Причинно-наслідкове моделювання (це аналітична техніка, зосереджена на причинно-наслідкових зв'язках).
- Імовірнісні графічні моделі (PGM виражає структуру умовної залежності між випадковими величинами).
- Матрична факторизація (це клас алгоритмів спільної фільтрації, які використовуються спеціально в системах рекомендацій).
- Ансамблеве навчання (техніка, що використовує кілька алгоритмів навчання для досягнення кращих результатів) [4].

Крім того, важлива частина їхньої системи рекомендацій покладається на A/B тестування. Вони постійно перевіряють різні параметри щодо пропозицій фільмів, ескізів і способу організації заголовків, щоб визначити, що викликає найбільший інтерес і зацікавленість. Наприклад, у випадку глядача, якому подобаються романтичні фільми, персоналізація ілюстрації може означати, що така особа побачить мініатюру, яка представляє романтичний аспект фільму.

Насправді все, що є на домашній сторінці, створено відповідно до уподобань користувача. Їх система рекомендацій ранжує назви таким чином, щоб представити їх у найкращому порядку. Це означає що:

- Існує кілька рядків пропозицій, які містять різні заголовки, які можна переглянути.
- Кожна назва ранжується в рядку.
- Кожна мініатюра також адаптована до конкретного користувача [5].

По суті, це набір алгоритмів, які використовують машинне навчання для аналізу даних користувачів і рейтингів фільмів. Щоб підвищити ефективність, Netflix створив 1300 кластерів рекомендацій на основі уподобань користувачів щодо перегляду. У результаті кожного разу, коли користувач вмикаєте Netflix, він бачить список фільмів і телешоу, адаптованих до його інтересів і профілю

користувача. Мета полягає в тому, щоб допомогти кожному користувачеві знайти фільм або телешоу, які йому сподобаються, і зробити це якомога швидше (за оцінками, у нього є лише 90 секунд для цього). Сама система рекомендацій залишається у серверній частині та невидима для користувача.

1.1.2. Рекомендаційна система Amazon

Наразі Amazon використовує спільну фільтрацію по елементам, яка масштабується до масивних наборів даних і створює високоякісні системи рекомендацій у режимі реального часу. Ця система є різновидом системи фільтрації інформації, яка прагне передбачити «рейтинг» або вподобань, які цікавлять користувача.

Рекомендації продукту, адаптовані до користувача, з більшою ймовірністю призведуть до більшої конверсії. Рекомендовані продукти становлять 35% доходу Amazon. Крім того, користувачам потрібні рекомендації схожих товарів, щоб допомогти відкрити нові продукти або порівняти товари.

Система рекомендацій Amazon здатна розумно аналізувати та передбачати переваги покупців, щоб запропонувати їм список рекомендованих продуктів.

Хоча в останні роки інші магазини також запровадили подібні функції на своїх веб-сайтах, система рекомендацій Amazon вважається однією з найкращих на ринку. Разом із розробкою технології штучного інтелекту Amazon почав працювати над алгоритмом, який зможе аналізувати товари, розміщені користувачами, і визначати купівельні переваги окремих клієнтів.

Алгоритм Amazon − це система рекомендацій, яка складається з кількох важливих частин, які відповідають за аналіз різних даних. Це можливо завдяки технології, заснованій на штучному інтелекті та машинному навчанні

Алгоритм Amazon вибирає рекомендовані продукти для кожного користувача на основі їхніх попередніх покупок, взаємодії та оцінок інших

представлених товарів і поєднує їх із подібними товарами, які переглядають користувачі зі схожими вподобаннями та інтересами.

Алгоритм рекомендацій Amazon є ключовим елементом використання ШІ для покращення персоналізації веб-сайту. Надаючи рекомендації, які максимізують потенційну цінність для окремих клієнтів, Amazon може підтримувати зацікавленість споживачів і пропонувати їм цікаві продукти, про які вони навіть не думають.

Для персоналізації покупок компанія створила інструмент Amazon Personalize – це сервіс штучного інтелекту та машинного навчання, який спеціалізується на розробці рекомендаційних системних рішень. Він автоматично аналізує дані, вибирає функції та алгоритми, оптимізує модель на основі даних, а також впроваджує та підтримує модель для створення рекомендацій у реальному часі [6].

Ця система виходить за рамки стандартних рекомендацій електронної комерції та може використовуватися розробниками для створення складних інтелектуальних систем, які рекомендують інші веб-сайти.

Крім самих рекомендацій, Amazon використовує кілька інших алгоритмів штучного інтелекту, що відповідають за підтримку різних аспектів роботи платформи.

Компанія також використовує фірмовий алгоритм A9, який відповідає за інтелектуальний пошук товару на сайті. Алгоритм A9 аналізує та класифікує окремі бренди та їхні продукти на платформі, завдяки чому він може пропонувати клієнтам Amazon релевантні та персоналізовані результати пошуку. Ця система також є основою для визначення того, які продавці відображатимуться покупцям на головній сторінці.

Алгоритм Amazon A9 базується на 3 основних принципах роботи:

- Щоб визначити найкращі продукти, він враховує ключові слова,

- вміст, дані продавців, думки та відгуки, а також показники повернення.
- Алгоритм A9 класифікує товари на основі історії результатів продажів, точності збігу тексту, ціни та наявності на складі у окремих продавців.
- Існують також непрямі фактори, які впливають на позицію окремих продуктів у класифікації алгоритму. Найважливіші з них включають, серед іншого, варіанти доставки та оплати, описи та фотографії продуктів, преміум-контент, рекламу та акції [7].

Amazon нещодавно оновив алгоритм A9, який тепер називається алгоритмом A10. Оновлення змінило багато аспектів його роботи, зробивши його більш зосередженим на поведінці покупця, ніж на характеристиках продукту.

Щоб надавати клієнтам точні рекомендації щодо продукту, алгоритм Amazon повинен аналізувати величезні обсяги даних. Таким чином, він краще розуміє поведінку всіх користувачів та інтереси кожного глядача.

Для цього система рекомендацій збирає два типи інформації:

- Загальні дані про товари та користувачів;
- дані про зв'язки і залежності між ними [8].

Знайомство з існуючими відносинами в інтернет-магазині надасть системі рекомендацій розуміння реальних механізмів, що керують рішеннями клієнтів про покупку.

Окрім збору інформації про стосунки та зв’язки, алгоритм рекомендацій Amazon також використовує різні типи даних про продукти та користувачів:

- Дані про поведінку користувачів – цей тип даних є корисною інформацією про вподобання окремих клієнтів, їхню взаємодію з відповідними продуктами. Amazon використовує файли cookie для

збору даних про історію веб-перегляду, оцінки "подобається" або тривалість сеансу.

- Демографічні дані користувача – ці дані користувачів пов'язані з особистою інформацією про окремих клієнтів, як-от вік, освіта, дохід і місцезнаходження. Для збору таких даних необхідна згода користувача [9].

Одним із напрямків дослідження нових рекомендаційних алгоритмів є так звані Bandit-based алгоритми.

Алгоритм «бандитів» заснований на машинному навчанні шляхом посилення і намагається розробити можливістб продажу нових продуктів, використовуючи ті, що вже приносять прибуток. Алгоритми на основі бандитів також можна використовувати для вибору в режимі реального часу між кількома моделями рекомендацій на основі того, як користувачі реагують на різні пропозиції продукту.

Іншим інноваційним підходом до рекомендацій алгоритмів, над яким працювала Amazon, є алгоритм випадкових перешкод. В основному він зосереджений на виявленні факторів, які спонукали окремих клієнтів звернути увагу на конкретні продукти.

Розробивши алгоритм, який поєднує випадкове втручання з існуючими алгоритмами рекомендацій, дослідники Amazon змогли створити покращені рекомендації, взявши до уваги різні фактори, що вводять в оману.

1.1.3. Рекомендаційна система YouTube

Коли рекомендації YouTube найкращі, вони з'єднують мільярди людей у всьому світі з унікальним вмістом, який надихає, навчає та розважає. Рекомендації збільшують значну частину загальної аудиторії на YouTube, навіть більше, ніж підписки на канали чи пошук.

Система рекомендацій побудована на простому принципі допомоги людям у пошуку відео, які вони хочуть переглянути, і це принесе їм цінність. Знайти рекомендації можна в двох основних місцях: на домашній сторінці та на панелі «Далі». Домашня сторінка відображає суміш персоналізованих рекомендацій, підписок, а також останні новини та інформацію. Панель «Далі» з'являється, коли йде перегляд відео, і пропонує додатковий вміст на основі того, що зараз переглядається, а також інші відео, які, на думку користувача, можуть зацікавити.

У 2008 році, коли YouTube почали створювати систему рекомендацій, досвід був зовсім іншим. Припустимо, юзер здебільшого дивиться кулінарні відео. Чи не було б неприємно, якби домашня сторінка рекомендувала користувачу лише останні спортивні та музичні відео, оскільки вони мали найбільше переглядів? Це був YouTube на початку. Система ранжувала відео на основі популярності, щоб створити одну велику сторінку «Тенденції». Небагато людей переглянули ці відео, і більшість переглядів на YouTube прийшли через пошук або спільні посилання за межами платформи [10].

Рекомендації щодо "Далі" – для цього YouTube починає зі знання того, що кожен має унікальні звички перегляду. Потім система порівнює звички юзера перегляду з тими, які схожі на юзера, і використовує цю інформацію, щоб запропонувати інший вміст, який юзер можете переглянути. Отже, якщо юзеру подобаються тенісні відео, і система помічає, що інші, яким подобаються ті самі тенісні відео, що й ви, також подобаються джазові відео, вам можуть рекомендувати джазові відео, навіть якщо ви ніколи раніше не дивилися жодного (для категорій, як-от новини це може працювати інакше).

У системі використовуються дві нейронні мережі: одна для формування кандидатів, інша для рейтингу. Мережа створення кандидатів використовує події з історії активності користувача на YouTube, щоб отримати невелику підмножину (сотні) відео з великого корпусу. Ці кандидати мають бути дуже

точними, але водночас широко релевантними для користувача. Лише спільна фільтрація дозволяє мережі генерації кандидатів надавати широкі налаштування. Користувачі групуються на основі грубих характеристик, таких як ідентифікатори перегляду відео, маркери пошукового запиту та демографічні дані [11].

Щоб визначити відносну важливість серед кандидатів із сильним запам'ятовуванням, під час представлення кількох «найкращих» рекомендацій у списку потрібне представлення тонкого рівня. Мережа рейтингу досягає цієї мети, призначаючи оцінку кожному відео на основі визначеної цільової функції та великої кількості характеристик, які описують відео та користувача. Користувачеві показуються відео з найвищими балами, які ранжуються за їхніми балами.

Двоетапний підхід до рекомендацій дозволяє нам давати рекомендації з великого корпусу відео (мільйони), будучи впевненими, що обмежена кількість відео, які з'являються на пристрої, є персоналізованою та цікавою для користувача. Ця архітектура також дозволяє змішувати кандидати, надані з інших джерел, як-от описані в попередній статті [12].

Алгоритм YouTube покроково:

1. YouTube переглядає історію переглядів користувача, щоб скласти список рекомендованих відео (воронка «покоління кандидатів»).
2. У воронці «рейтингу» відео присвоюється оцінка на основі релевантності та безлічі інших факторів. Потім відео перераховуються в порядку від найвищого до найменшого і показані користувачеві [13].

Цей процес машинного навчання є важливою частиною алгоритму рекомендацій YouTube, оскільки їх головною метою є задоволення глядачів. Задоволеність глядачів вимірюється за допомогою відповідей на запитання та сигнали залучення, включаючи лайки, антипатії та кліки «не зацікавлені».

Згідно YouTube, платформа «намається показувати кожному глядачеві відео, які вони, швидше за все, переглядають і отримують задоволення». Це полягає в тому, щоб утримувати користувачів на YouTube як можна більше, що, природно, призводить до того, що вони бачать більше реклами.

Фактори рекомендацій включають в себе:

- Історія перегляду користувача: які теми, відео та канали він дивився і використовував у минулому?
- Відео по темі.
- Відео, які інші глядачі дивляться разом із поточним відео: що дивляться схожі користувачі?

1.2. Постановка задачі

Мета даної роботи полягає у розробці веб застосунку, який буде рекомендувати фільм юзеру, на основі його вподобань. Використовуючи різні алгоритми даних і висновки про те, що сподобається покупцеві на основі його минулих переваг або про те, що було куплено аналогічними клієнтами, рекомендаційні системи можуть систематично заохочувати додаткові витрати, пропонуючи набагато привабливіший досвід користувача і залучати нових клієнтів.

Першим етапом досягнення мети необхідно обрати метод рекомендацій контенту, який буде найкраще підходити для вирішення цієї заачі та найефективнішим. Оскільки методів рекомендаційних систем безліч потрібно провести аналіз наявних методів визначити їх переваги і недоліки.

Другим етапом необхідно обрати алгоритм рекомендаційних систем і як він буде працювати в різних ситуаціях. Також необхідно вирішити який вигляд буде мати веб застосунок, з яких елементів складатись та як користувачі будуть з ним взаємодіяти. Продумати процес авторизації, які дані про користувача необхідні будуть.

Третім етапом буде створення системи, яка буде створювати персональні рекомендації користувача, базуючись на його вподобаннях та відгуках. При авторизації користвач буде вводити персональні дані, які будуть зберігатись в системі. На основі відгуків, оцінок буде надаватись список рекомендацій. Після генерації, система запропонує декілька варіантів, й при виборі його користувачем, враховуватиме цей вибір для наступної генерації.

1.3. Висновки до розділу

У цьому розділі було проаналізовано наявні сервіси, які використовують рекомендаційні системи. Важливість рішень, заснованих на машинному навчанні та штучному інтелекті, для найбільших веб-сайтів в Інтернеті чітко демонструє, що наразі вони є однією з найважливіших тенденцій в онлайн-активності. Системи рекомендацій вмісту здатні довше привертати увагу користувачів, а веб-сайти, які впровадили комплексні системи рекомендацій, мають значну перевагу перед конкурентами. Для індустрії електронної комерції, що постійно розвивається, рішення, які покращують персоналізацію веб-сайту на основі машинного навчання, є дуже хорошим способом розвитку та максимізації прибутку.

Найбільші онлайн-платформи все частіше використовують штучний інтелект для покращення взаємодії з користувачем. Ці рішення є напрочуд ефективними та можуть також допомогти галузі електронної комерції.

Також була описана актуальність, сформована задача, яка буде проаналізована під час роботи над дисертацією. Було вказано призначення розробки рекомендаційної системи, цілі та задачі, які повинні бути досягнуті під час роботи з системою рекомендацій контенту.

2. ДОСЛІДЖЕННЯ МЕТОДІВ ТА ВИЗНАЧЕННЯ ПІДХОДУ ДЛЯ ПОБУДОВИ РЕКОМЕНАЦІЙНИХ СИСТЕМ

2.1. Типи данних в рекомендаційних системах

Рекомендаційні системи працюють із двома видами інформації:

- Характерна інформація. Це інформація про товари (ключові слова, категорії тощо) та користувачів (переваги, профілі тощо).
- Взаємодія користувача з елементом. Це така інформація, як рейтинги, кількість покупок, лайків тощо.

Найважливішим елементом у створенні системи рекомендацій є дані. Є три типи даних:

- Явні дані зазвичай мають форму числа (наприклад, 5-зірковий рейтинг), яке дає користувач продукту. Прикладами явних даних є оцінки продуктів, надані клієнтами на Amazon, або рейтинги курсів, надані користувачами на навчальній платформі Udemy. Цей тип даних важко збирати, оскільки вони вимагають додаткового введення від користувачів, тому потрібно більше часу, щоб отримати пул оцінок, достатньо широкий для створення корисної моделі машинного навчання.
- Неявні дані легко зібрати. Це будь-які дані, зосереджені на тому, як користувач взаємодіє з доступними продуктами/вмістом. Основна проблема з цим типом даних полягає в тому, як перетворити поведінку користувача в його уподобання. Але є ефективні способи це зробити. Прикладами неявних даних є кількість відтворення пісні на Spotify, кількість кліків за посиланнями на продукт або історія покупок на Amazon.
- Останнім типом даних є опис товару. Оскільки цей тип даних часто є неструктурованим (наприклад, має форму вільного тексту), нам потрібно впровадити деяку додаткову попередню обробку, щоб

отримати відповідну інформацію та помістити її в структуровану форму. Прикладами описів продукту є список акторського складу фільму на Netflix, автор пісень на Spotify або опис продукту на Amazon [14].

2.2. Можливі проблеми рекомендаційних систем

Можливості збору інформації, що надаються Інтернетом, істотно полегшили використання "мудрості натовпу" за допомогою колаборативної фільтрації. Але дуже велика кількість доступних даних ускладнює реалізацію цієї можливості. Наприклад, поведінка деяких користувачів добре піддається моделюванню, але інші користувачі не демонструють сталої поведінки. Велика кількість таких користувачів може призводити до зниження точності результатів рекомендаційної системи та до зменшення її ефективності. Крім цього, користувачі можуть задіяти рекомендаційну систему для підвищення значущості одного товару щодо іншого товару – до наприклад, за допомогою відправки позитивних відгуків про один товар і негативних відгуків про його конкурентів. Добре спроектована рекомендаційна система має справлятися з цими проблемами.

Існує ще одна проблема, властива великим рекомендаційним системам, пов'язана вона з масштабованістю. Базові алгоритми добре працюють із порівняно невеликими обсягами даних, але зі збільшенням цих наборів отримання результатів на колишньому рівні якості за допомогою базових алгоритмів може стати проблемою. У разі офлайнової обробки це може не складати великої проблеми, але для сценаріїв реального часу потрібні більш спеціалізовані підходи. Деякі проблеми породжує необхідність дотримання правил конфіденційності. Рекомендаційні алгоритми здатні розпізнавати різні закономірності, про існування яких можуть навіть підозрювати. Приклад такої ситуації мав місце у великій компанії, яка спромоглася обчислювати індекс

прогнозування вагітності на основі купівельних переваг. Після отримання цільових рекламних оголошень батько дочки-підлітка із подивом дізнався про її вагітність. Прогнозуюча система цієї компанії виявилася настільки точною, що змогла передбачити очікувану дату пологів у майбутньої мами на основі товарів, що купуються нею [15].

2.3. Методи для побудови рекомендаційних систем

2.3.1. Content-Based Filtering

Методи фільтрації на основі вмісту базуються на описах продукту та вподобаннях користувача. Цей тип системи рекомендує продукти, схожі на продукти, які подобалися користувачеві в минулому. Цей тип системи рекомендацій базується на трьох етапах:

Аналізатор опису продукту – на цьому етапі опис продукту аналізується за допомогою техніки виділення ознак, щоб перетворити вихідний опис у вектор товару. Використовуючи вектори елементів, система обчислює подібність між продуктами.

Аналізатор профілю користувача – на другому кроці система збирає налаштування користувача, дані історії користувача та створює профіль користувача, який представлений вектором користувача. Функції, описані у векторі користувача, збігаються з функціями у векторах елементів [16].

Фільтрація компонентів – на останньому кроці система вибирає рекомендації на основі векторів користувача та елементів (наприклад, за допомогою косинусної подібності).

Тут вміст стосується атрибуту продукту, який подобається користувачеві. У системі цього типу продукти позначаються певними ключовими словами, потім система намагається зрозуміти, що хоче користувач, переглядає свою базу даних і, нарешті, намагається рекомендувати різні продукти, які хоче користувач.

Давайте візьмемо приклад системи рекомендацій фільмів, де кожен фільм пов'язаний зі своїми жанрами, які в наведеному вище випадку називаються атрибутами. Тепер припустимо, що приходить користувач А, і система спочатку не має жодних даних про користувача. Тому спочатку вона намагається рекомендувати популярні фільми користувачам або намагається отримати деяку інформацію про користувача, використовуючи форму, заповнену користувачем. Через деякий час збирається інформація про користувача – він дає хорошу оцінку фільмам у жанрі бойовик і погану оцінку фільмам у жанрі аніме. Тому система рекомендує користувачу бойовики. Але ми ж неможемо сказати, що користувачеві не подобаються анімаційні фільми, оскільки, можливо, користувачеві не подобається цей фільм через якусь іншу причину, як-от акторська гра чи історія, але насправді йому подобаються анімаційні фільми, і в цьому випадку системі потрібні додаткові дані [17].

Присвоєння атрибутів може бути величезною справою. Багато компаній вдаються до використання груп експертів із певної тематики, щоб призначити атрибути кожному елементу вручну. Наприклад, Netflix найняв сценаристів, щоб вони оцінювали шоу за різними аспектами: від місць зйомки та акторів до сюжетних ліній, тону та емоційних ефектів. Отримані теги, які використовуються рекомендувачем, алгоритмічно об'єднуються для групування фільмів, які мають подібні аспекти.

Профілі користувачів є ще одним важливим елементом систем рекомендацій на основі контенту. Профілі включають об'єкти бази даних, з якими користувач взаємодіяв − купував, переглядав, читав, дивився чи слухав− а також призначені їм атрибути.

Атрибути, які з'являються в кількох об'єктах, мають більшу вагу, ніж ті, які з'являються рідше. Це допомагає встановити ступінь важливості, оскільки не всі атрибути об'єкта однакові для користувача. Відгуки користувачів також мають важливе значення під час зважування елементів, тому веб-сайти, які

надають рекомендації, постійно просять вас оцінити продукти, послуги чи вміст.

На основі вагових коефіцієнтів атрибутів та історії система рекомендацій створює унікальну модель уподобань кожного користувача. Модель складається з атрибутів, які можуть знадобитись або ні користувачеві на основі минулих дій, зважених за важливістю. Моделі користувачів порівнюються з усіма об'єктами бази даних, яким потім призначаються бали на основі їх схожості з профілем користувача.

Переваги:

- Модель не потребує даних інших користувачів, оскільки рекомендації стосуються конкретного користувача.
- Цей підхід полегшує масштабування для великої кількості користувачів.
- Модель може охопити конкретні інтереси користувача та може рекомендувати елементи, які цікавлять дуже мало інших користувачів.
- Нові елементи можуть бути запропоновані до того, як їх оцінить велика кількість користувачів, на відміну колективної фільтрації.
- Фільтрування на основі вмісту також є цінним для підприємств із великими бібліотеками, що містять один тип продукту, наприклад смартфони, де рекомендації мають базуватися на багатьох окремих функціях.
- Високорелевантні рекомендації створюють відчуття відкритості для користувача, підвищуючи рівень його довіри до запропонованих рекомендацій. Для порівняння, із спільною фільтрацією ймовірні випадки, коли користувачі не розуміють, чому вони бачать певні рекомендації. Наприклад, скажімо, група користувачів, які придбали парасольку, також придбали пуховики. Спільна система може

рекомендувати пухові пуховики іншим користувачам, які купили парасолі, але не цікавляться ними та ніколи не переглядали та не купували цей продукт.

- Уникнення проблеми «холодного запуску». Спільна фільтрація створює потенційний сценарій холодного запуску, коли на новому веб-сайті чи спільноті мало нових користувачів і відсутні зв'язки користувачів. Хоча фільтрація на основі вмісту потребує певних початкових вхідних даних від користувачів, щоб почати давати рекомендації, якість ранніх рекомендацій, як правило, краща, ніж система спільної роботи, яка вимагає додавання та кореляції мільйонів точок даних перед оптимізацією.

Недоліки:

- Модель може давати рекомендації лише на основі наявного інтересу користувача. Іншими словами, модель має обмежену здатність розширювати існуючі інтереси користувача.
- Масштабованість – це виклик. Кожного разу, коли додається новий продукт, послуга або новий вміст, його атрибути повинні бути визначені та позначені тегами. Важкий, нескінченний характер призначення атрибутів може ускладнити масштабованість і забрати багато часу.
- Атрибути можуть бути неправильними або суперечливими. Рекомендації на основі вмісту настільки ж хороші, наскільки хороші експерти з певної теми, які позначають елементи. Потенційно мільйонам елементів може знадобитися призначення атрибутів, і оскільки атрибути можуть бути суб'єктивними, багато з них можуть бути неправильно позначені. Процес, який забезпечує послідовне й точне застосування атрибутів, має першочергове значення. В іншому

випадку система рекомендацій на основі вмісту не працюватиме належним чином [18].

2.3.2. Collaborative filtering

Метод спільної фільтрації заснований на збиранні та аналізі даних про поведінку користувача. Це включає дії користувача в Інтернеті і прогнозування того, що йому сподобається, на основі подібності з іншими користувачами. Наприклад, якщо користувачеві А подобаються яблука, банани і виноград, а користувачеві В подобаються яблука, банани і гранат, у них схожі інтереси. Таким чином, ймовірно, що А захоче гранат, а Б – виноград. Так відбувається спільна фільтрація.

Щоб побудувати систему, яка може автоматично рекомендувати елементи користувачам на основі вподобань інших користувачів, першим кроком є пошук схожих користувачів або предметів. Другим кроком є прогнозування рейтингів елементів, які ще не оцінені користувачем.

Такі підходи, засновані на пам'яті, безпосередньо працюють із значеннями записаних взаємодій або даними і, по суті, базуються на пошуку найближчих сусідів, тобто пошуку найближчих користувачів від користувача, який вас цікавить, і пропонують найпопулярніші елементи серед цих сусідів. Створена модель передбачає припущення, що в основі лежить розуміння, яке пояснює взаємодію між користувачем і елементом, і намагається виявити його, щоб зробити нові прогнози. Він рекомендує елемент користувачеві А на основі інтересів подібного користувача В. Для методу спільного фільтрування не потрібні характеристики елементів, які потрібно надати. Кожен користувач і елемент описуються вектором ознак або вбудовуванням [19].

Одна важлива річ, про яку слід пам'ятати, полягає в тому, що в підході, заснованому суто на спільній фільтрації, подібність не обчислюється за такими факторами, як вік користувачів, жанр фільму або будь-які інші дані про

користувачів або елементи. Він обчислюється лише на основі оцінки (явної чи прихованої), яку користувач дає елементу. Наприклад, двох користувачів можна вважати схожими, якщо вони дають однакові оцінки десяти фільмам, незважаючи на велику різницю у віці.

Більшість систем спільної фільтрації застосовують так звану техніку на основі індексу подібності. У підході на основі сусідства кількість користувачів вибирається на основі їх схожості з активним користувачем. Висновок для активного користувача робиться шляхом обчислення середньозваженого рейтингу вибраних користувачів.

Системи спільної фільтрації зосереджені на стосунках між користувачами та елементами. Подібність елементів визначається подібністю оцінок цих товарів користувачами, які оцінили обидва елементи.

Існує два типи методу спільної фільтрації:

- Спільна фільтрація на основі пам'яті: один із методів, який обчислює подібність між користувачами або елементами, використовуючи попередні дані користувача на основі рейтингу. Основна мета цього методу – описати ступінь схожості між користувачами чи об'єктами та виявити однорідні оцінки, щоб запропонувати приховані елементи. Підходи, засновані на пам'яті, безпосередньо працюють зі значеннями записаних взаємодій, не припускаючи відсутності моделі, і в основному засновані на пошуку найближчих сусідів (наприклад, знаходять найближчих користувачів від користувача, що цікавить, і пропонують найпопулярніші елементи серед цих сусідів). Підходи, що базуються на моделях, припускають базову «генеративну» модель, яка пояснює взаємодію користувача з елементом і намагається виявити її, щоб зробити нові прогнози.

Метод складається з наступних двох методів:

- User-based: відомий той самий користувач A, який має подібні рейтинги для однорідних елементів користувача B. Далі метод вказує на предмет користувача A, на який користувач B ніколи не посилався.
- Item-based: на основі елементів ми знаходимо ті самі елементи, які цільовий користувач уже переглядав. Спочатку визначаємо цільового користувача, далі знаходимо відповідні елементи, які мають ті самі оцінки, що й цільовий користувач. Наступним кроком прогнозуємо рейтинги для тих самих елементів. Якщо прогнозовані рейтинги вищі за порогове значення, запропонуйте їх цільовому користувачеві [20].

У системі, де більше користувачів, ніж елементів, фільтрація на основі елементів є швидшою та стабільнішою, ніж на основі користувачів. Це ефективно, оскільки зазвичай середня оцінка, яку отримує товар, не змінюється так швидко, як середня оцінка, яку користувач дає різним елементам. Також відомо, що він ефективніший, ніж підхід, орієнтований на користувача, коли матриця оцінок розріджена. Хоча підхід на основі елементів погано працює для наборів даних із елементами, пов'язаними з переглядом або розвагами, такими як YouTube, де рекомендації, які він дає, здаються дуже очевидними для цільових користувачів. Такі набори даних дають кращі результати за допомогою методів матричної факторизації.

Спільна фільтрація на основі моделі:

Переваги:

- Система може точно рекомендувати складні предмети, не розуміючись на предметі.
- Не має ніякої залежності від машинно-аналізованого контенту.
- Він адаптивний. Система фіксує зміни в інтересах користувачів.

- Покупці можуть ширше ознайомитися з багатьма різними продуктами, що створює можливості для заохочення покупців до постійних покупок продуктів.

Недоліки:

- «Холодний запуск» для нових елементів, які додаються до списку. Доки хтось не оцінить їх, вони не будуть рекомендовані.
- Розрідженість даних може вплинути на якість рекомендацій на основі користувачів, а також посилити проблему холодного запуску, згадану вище.
- Масштабування може бути проблемою для зростаючих наборів даних, оскільки складність може стати занадто великою. Якщо набір даних великий, рекомендації на основі елементів працюють швидше, ніж на основі користувачів.
- Можлива ситуація, коли вже і так популярні елементи рекомендуються, а елементи з розділу довгого хвоста можуть ігноруватися.
- Синоніми. Спільна фільтрація не може визначити синоніми. Тут «синоніми» стосуються подібних предметів, позначених або названих по-різному. Спільна фільтрація не може виявити прихований зв'язок між синонімами, тому ці продукти розглядатимуться по-різному. Наприклад, здавалося б, різні предмети «Рюкзак» і «Портфель» насправді стосуються одного предмета [21].

Оскільки розрідженість і масштабованість є двома найбільшими проблемами для стандартного методу, з'являється більш просунутий метод, який розкладає вихідну розріджену матрицю на матриці низької розмірності з прихованими факторами/функціями та меншою розрідженістю. Це матрична факторізація. Кількість прихованих факторів впливає на рекомендації таким

чином, що чим більша кількість факторів, тим персоналізованішими стають рекомендації. Але занадто багато факторів можуть призвести до переобладнання моделі [22].

2.3.3. Метод демографічної фільтрації

Оскільки системи рекомендацій використовують дані користувача для надання рекомендацій, основним бар'єром для всіх типів систем рекомендацій є незначний обсяг даних користувача. Незначна кількість даних значно знижують точність рекомендаційних систем, тому вивчити вплив обмежених даних і як це можна вирішити вкрай важливо.

Коли новий продукт додається в систему, він не має ніякого відношення до інших продуктів і користувачів. Ця нестача даних особливо складна для систем спільної фільтрації, які використовують зв'язки між елементами та користувачами, тоді як фільтрація на основі вмісту класифікує предмет на основі його характеристик.

Проблема холодного запуску виникає, коли створюється новий користувач або коли існуючий користувач не надав достатньо явних даних, щоб задовольнити систему рекомендацій. Відсутність даних призводить до рекомендацій, які не відповідають вподобанням користувача. Щоб вирішити проблему з новим користувачем, деякі системи, такі як Netflix, просять користувачів надати параметри під час реєстрації.

Демографічна фільтрація пропонує користувачам зі схожим демографічним походженням схожі фільми, популярні та добре оцінені незалежно від жанру чи будь-яких інших факторів. Тому, оскільки він не враховує індивідуальні смаки кожної людини, він забезпечує простий результат, але його легко реалізувати [23].

2.3.4. Нейронна колаборативна фільтрація

Нейронна колаборативна фільтрація – це використання нейронної мережі певної архітектури для моделювання на основі наявної інформації власних векторів об'єктів та користувачів, а також вивчення функції, що описує їхню взаємодію, на основі якої складаються рекомендації.

Нехай є інформація про M користувачів та N фільмів, отже отримаємо матрицю взаємодій $Y \in R^{M*N}$, де для користувача u та фільму i елемент $y = 1$, якщо користувач переглянув цей фільм, та інакше $y = 0$ [24].

Першим кроком у цю модель подаються два різних вектори, що характеризують відповідно фільм і користувача. Припустимо використовувати простий варіант бінарного вектора довжини M для користувача і N – для фільму, в якому компонента, що дорівнює 1, один раз зустрічається на місці індексу об'єкта, який цей вектор описує. Це так зване one-hot кодування. Описані вектори надходять парою користувач-фільм на вхід нейронної мережі. Далі вони окремо проходять через незалежні нейронні рівні “embedding” для перетворення та ущільнення. На виході після цього отримуємо власні вектори фільму та користувача. Відповідно, на цьому етапі нейронної мережі відбувається навчання для отримання певного векторного представлення даних, тобто переведення вхідних даних (one-hot вектор фільму і користувача) в простір ознак заданої довжини. Отримані власні вектори фільму та користувача об'єднуються і надходять у наступний повнозв'язковий нейронний шар. Далі, архітектура мережі являє собою повнозв'язкові рівні, кількість яких може змінюватись, а кількість нейронів зменшується зі збільшенням рівня. Загальну структуру моделі для нейронної колаборативної фільтрації зображено на рис. 2.1.

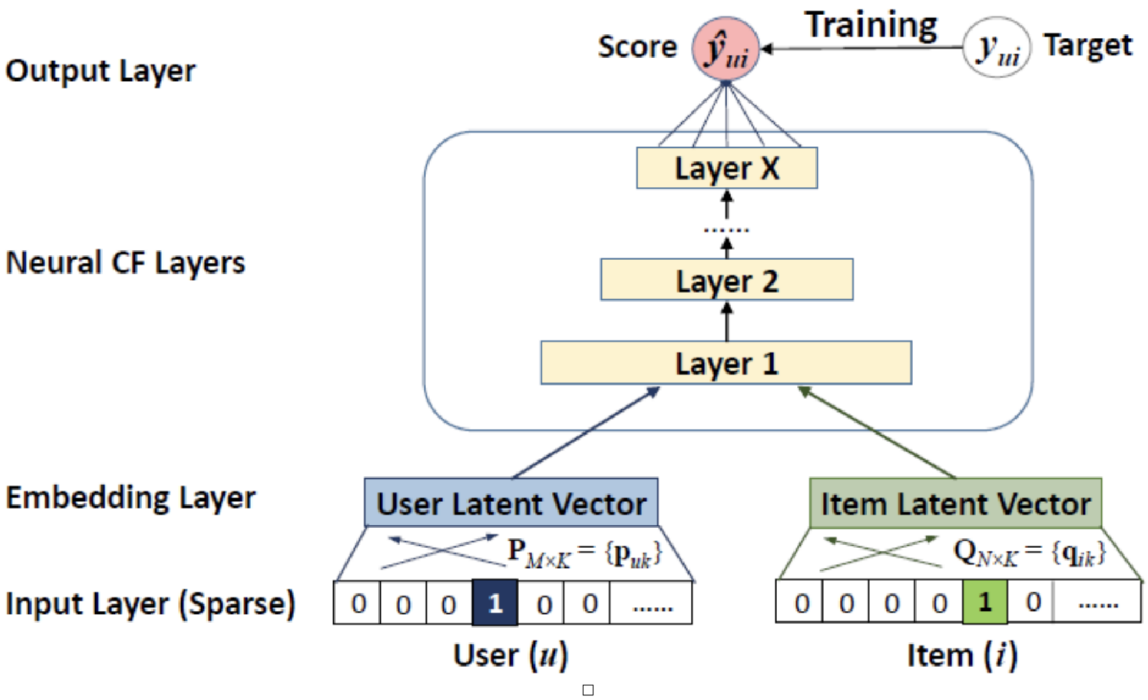


Рис. 2.1. Загальна схема нейронної колаборативної фільтрації

Останній рівень, або вихід мережі, є однонейронним повнозв'язковим рівнем з логістичною функцією активації. Відповідно, як відповідь мережа видає ймовірність взаємодії користувача з фільмом в діапазоні від 0 до 1. Навчається мережа за допомогою стохастичного градієнтного спуску. Мета навчання − мінімізації функції помилки між наявними значеннями матриці Y та передбаченими $\hat{y}$. Під час навчання використовуються як позитивні приклади, тобто переглянуті фільми, а й негативні − фільми, з яким конкретний користувач ніяк не взаємодіяв [25].

У цій роботі нейронна мережа навчатиметься за допомогою навчання з учителем. При цьому методі навчання на вхід нейронної мережі подається не тільки безпосередньо вхідні дані, тобто інформація про фільм і користувача, а й очікувана відповідь мережі, у разі поставленого завдання оцінка фільму користувачем. Для реалізації навчання з учителем готується спеціальний набір даних, який називається відповідно навчальним. Його основна особливість у

тому, що він складається не тільки з підготовлених вхідних даних для нейронної мережі, але до кожної одиниці навчального набору даних додається реальне відповідне реальне значення, яке називається міткою, яке очікується як вихідне значення мережі після навчання. Також частина навчальних даних може не брати участь у навчанні, тоді вона називається тестовою і використовується для оцінки якості моделі та розрахунку метрик. Період навчання нейронної мережі, за який усі навчальні набори виявляться на вході мережі рівно один раз, називається епохою.

Отже, метою та результатом навчання є нейронна мережа, в результаті роботи якої її вихідні значення максимально наближені до навчальних міток. Навчання відбувається до тих пір, поки не будуть досягнуті певні значення метрик якості та функції помилки, що показують, що мети досягнуто.

Одним із найпоширеніших алгоритмів навчання є методи першого порядку. У цьому роботі використовуватиметься метод зворотного поширення помилки. Для його застосування необхідно розрахувати помилку нейронної мережі, тобто різницю мітки та вихідного значення, наприклад, за методом найменших квадратів. Також, йому необхідні обчислення похідних функції активації. Відповідно, сам перерахунок ваги відбувається за обраною варіації методу градієнтного спуску тобто зменшення помилки мережі досягається шляхом зміни ваг за принципом руху в зворотний бік від розрахованого градієнта функцій активації.

2.3.5. Гібридний підхід

Суміш методів колаборативної фільтрації та методів на основі вмісту під час надання пропозицій повинен враховуватись контекст фільму. Відношення користувача до елемента та відношення користувача до користувача також відіграють важливу роль під час рекомендації. Цей фреймворк дає рекомендації щодо фільмів відповідно до знань користувача, надає унікальні

рекомендації та вирішує проблему, якщо конкретний покупець ігнорує відповідні дані. Дані профілю користувача збираються з веб-сайту, контекст фільму також враховує перегляд фільму користувачем і дані партитур фільму.

Дані складаються з узагальнення подібних розрахунків. Цей метод називається гібридним підходом, у якому обидва методи використовуються для отримання результатів. Якщо порівнювати цю систему з іншими підходами, ця система має вищу точність пропозицій. Основною причиною є відсутність інформації про доменні залежності фільтрації та інтерес людей до контентної системи.

Коли ці два підходи працюють разом, ви отримаєте більше знань, що призведе до кращих результатів; він досліджує нові шляхи до значного основного вмісту та методів спільної фільтрації з даними про поведінку покупців.

Ця система реалізувала обидві системи та подолала більшість слабких місць алгоритмів кожної системи та покращила продуктивність системи. Методи класифікації та кластеризації використовуються для отримання кращих рекомендацій, що підвищує точність і точність. Наш метод може бути довшим, ніж інші правила, щоб рекомендувати відео, пісні, журнали, місце проведення, сайт електронної комерції, туризм тощо.

2.4. Вибір методу для побудови рекомендаційної системи

В ході написання роботи було обрано гібридний метод побудови рекомендаційних систем. Для вирішення проблеми холодного запуску для нових користувачів платформи будемо використовувати метод демографічної фільтрації. Якщо вже є якісь данні тоді будемо використовувати метод на основі вмісту та метод нейронної колаборативної фільтрації. Таким чином ми вирішимо проблему обмеженості розширення інтересу в методі на основі вмісту, а також наш алгоритм буде більш адаптивним, якщо раптом зміняться

ітереси користувача. Вирішити проблеми методу колаборативної фільтрації, а саме те, що рекомендуються і так популярні фільми та проблему розрідженності даних допоможе метод на основі вмісту.

Тобто для більш персональних рекомендацій, аналізуючи його профіль та поведінку користувача в минулому будемо використовувати метод на основі вмісті. А метод колаборативної фільтрації буде використаний для користувачів з однаковим стилем навчання. Далі буде побудовано нейронну мережу з учителем для рекомендації контенту.

Усе, що буде робити користувач у веб застосунку буде записуватись у базу даних. Це може бути обрання контенту, залишення відгуку та оцінки, перегляд картки фільму. Кожну дію буде розділено на тип дії, яку зробив користувач, а також буде записано час дії. Ці данні будуть використані для реалізації методу.

2.5. Метрики ефективності побудованої моделі

2.5.1. Метрики точності

Нехай

d'_{iu} – прогнозований рейтинг фільму i від користувача u,

d_{iu} – існуючий рейтинг фільму i від користувача u,

n – кількість рейтингів.

Точність системи рекомендацій зазвичай оцінюють трьома основними показниками: кореневою середньоквадратичною помилкою MSE (2.1), середньоквадратичною помилкою RMSE (2.2) і середньою абсолютною помилкою MAE (2.3). Обидва варіанти вдалі, оскільки їх легко тлумачити: вони обидва за тією ж шкалою, що й оригінальні рейтинги. Однак один може бути кращим для використання, залежно від контексту набору даних [26].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{d'_{iu}\in n}(d_{iu} - d'_{iu})^2} \qquad (2.1)$$

$$\text{MSE} = \frac{1}{n}\sum_{d'_{iu}\in n}(d_{iu} - d'_{iu})^2 \quad (2.2)$$

$$\text{MAE} = \frac{1}{n}\sum_{d'_{iu}\in n}|d_{iu} - d'_{iu}| \quad (2.3)$$

MAE не дає жодного зміщення до екстремумів у термінах помилки. Якщо є викиди або великі похибки, програма буде порівнювати їх з іншими прогнозами. Таким чином, MAE слід віддати перевагу, дивлячись на точність рейтингу, коли насправді не треба дивитись на важливість викидів. Середньоквадратична помилка суворіше карає великі помилки

Одна з тенденцій середньоквадратичної помилки полягає в тому, що вона має тенденцію непропорційно штрафувати великі помилки, оскільки залишкова величина (член помилки) зведена в квадрат. Це означає, що RMSE більш схильний до впливу викидів або поганих прогнозів [28].

За визначенням, RMSE ніколи не буде таким малим, як MAE. Однак, якщо значення помилки слідують нормальному розподілу, використання RSME дозволяє реконструювати набір помилок, враховуючи достатню кількість даних. З іншого боку, MAE може точно відтворити лише 0,8 набору даних. Крім того, RSME не використовує абсолютні значення, що набагато зручніше з математичної точки зору під час обчислення відстані, градієнта чи інших показників. Ось чому більшість функцій витрат у машинному навчанні уникають використання MAE і радше використовують суму квадратичних помилок або середньоквадратичну помилку [29].

Набори даних для рекомендаційних систем часто містять кілька елементів, які містять найбільшу кількість оцінок, тоді як більшість елементів мають дуже мало оцінок. Це впливає на точність вимірювань, оскільки прогнозована точність для популярних елементів зазвичай відрізнятиметься від тих, що мають низький рейтинг. Рішення цієї проблеми полягає в тому, щоб при обчисленні RSME або MAE надати конкретні ваги для кожного

товару, які визначаються продавцем. Це може дозволити більш точне уявлення про те, як система рекомендацій оцінює всі елементи.

Переваги:

- Метрики прогнозної точності оцінюють точність фактичних прогнозів.
- Використовуючи показники точності прогнозування, можна впорядкувати всі прогнози.
- Усі показники точності прогнозування можна легко обчислити.
- Метрики точності добре відомі статистичні властивості, які дозволяють визначити значущість тестування відмінностей у точності різних рекомендаційних систем.

Недоліки:

- Системи рекомендацій, які видають результати користувачам, зазвичай видають ранжовані результати або просто набір рекомендацій. Точність (або неточність) фактичних рейтингів може бути неправильним способом вимірювання успіху рекомендації.
- Рейтингові системи з низькою деталізацією рейтингових шкал не можуть вимагати абсолютно правильних прогнозів [30].

2.5.2. Метрики точності класифікації

Показники точності класифікації застосовуються до оцінювання систем рекомендацій, які приймають детальні рішення щодо пар користувач–елемент: наприклад, Рекомендувати / Не рекомендувати. Метрики оцінюють частоту прийняття системою правильних / неправильних рішень.

У багатьох програмах система рекомендацій не прогнозує вподобаня користувачів щодо елементів, таких як рейтинги фільмів, а намагається рекомендувати користувачам елементи, які вони можуть використовувати. Наприклад, коли фільми додаються до черги, Netflix пропонує набір фільмів,

які також можуть бути цікавими, враховуючи доданий фільм. У цьому випадку нас цікавить не те, чи правильно система прогнозує їх рейтинги фільмів, а скоріше, чи правильно система прогнозує, що користувач додасть ці фільми до черги (використання елементів).

Під час автономної оцінки передбачення використання ми зазвичай маємо набір даних, що складається з елементів, які використовував кожен користувач. Потім ми вибираємо тестового користувача, ховаємо деякі з його виборів і просимо рекомендувача передбачити набір елементів, які використовуватиме користувач [31].

Тоді ми маємо чотири можливі результати для рекомендованих і прихованих елементів, як показано в таблиці 2.1.

Таблиця 2.1

Класифікація можливого результату рекомендації товару користувачеві

	Рекомендуємо	Не рекомендуємо
Релевантні	Істинно–позитивний (tp)	Хибно–Негативний (fn)
Не релевантні	Хибно–позитивний (fp)	Істинно–негативний (tn)

Ми можемо порахувати кіл-ькість прикладів, які потрапляють у кожну клітинку таблиці та обчисліть такі величини:

Точність – це частка всіх рекомендованих релевантних елементів.

$$Presicion = \frac{tp}{tp + fp} \tag{2.4}$$

Повнота – це частка всіх відповідних предметів, які були рекомендовані (2.5)

$$Recall = \frac{tp}{tp + fn} \tag{2.5}$$

Повнота та точність вимірюють різні аспекти точності системи рекомендацій. Їх можна поєднати в одну величину (2.6)

$$F - \text{міра} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.6)$$

Частка правильно класифікованих об'єктів, або Accuracy (2.7):

$$A = \frac{tp + tn}{tp + tn + fp + fn} \quad (2.7)$$

Коли класів більше, ніж 2, то дані метрики розраховуються через суми кожного класу чисельників і знаменника. Наприклад, також використовується далі метрика *mAP*, або mean average precision, розраховується за такою формулою (2.8):

$$mAP = \frac{\sum_{i=1}^{N} Precision_i}{N} \quad (2.8)$$

де N – кількість класів у завданні, а *Precision* – точність для класу i [32].

ROC криві вимірюють здатність системи фільтрації інформації відрізнити сигнал (відповідні пари користувач–елемент) від шуму (елементи, які не мають значення для користувачів).

Припустимо, що існує розподіл ймовірностей, пов'язаний із прогнозованим рівнем релевантності для релевантних і нерелевантних елементів. Чим краща система, чим більше відрізняються два розподіли ймовірностей.

Криві ROC будуються наступним чином.

- Розташовуються всі рекомендації за рейтинговою оцінкою.
- Для кожної граничної точки рейтингу (така ж, для кожної позиції в рейтинговому списку) рахується:
    - Відкликання (2.9);

$$Fallout = \frac{fp}{tp+fp} \quad (2.9)$$

- Частота помилкових позитивних результатів (2.10)

$$False\ Positive\ Rate = \frac{fp}{tp+tn} \quad (2.10)$$

Як правило, ми можемо очікувати компромісу між цими кількостями – якщо дозволити довші списки рекомендацій, як правило, покращує запам'ятовування, це також може зменшити точність. У програмах, де кількість рекомендацій, які можуть бути представлені користувачеві, визначена заздалегідь, найбільш корисним показником інтересу є Precision [33].

Переваги:

- Підходить для практичної оцінки емпіричних систем. Ці показники можуть бути використані для оцінки фактичної продуктивності рекомендаційної системи.
- Точність, відкликання, F-міра, ROC – це добре встановлені міри з відомими властивостями.
- ROC є надійним одночисловим показником.

Недоліки:

- Точність прогнозних заходів вимагає знання фактичних рейтингових значень. Однак класифікаційні заходи прогнозування вимагають знання чи визнали конкретну рекомендацію актуальною кінцевим споживачем. Це може бути важко отримати.
- Потреба у великому наборі даних. Ці заходи можуть вимагати оцінки в цілому наборі дат, щоб дійсно забезпечити хорошу роботу.
- Тоді як криві ROC дозволяють спостерігати вплив порядку ранжування рекомендацій, обміни в ранжуванні часто зберігають значення [34].

2.5.3. Метрики подібності

Similarity – це числова міра ступеня схожості двох елементів. Similarity часто є невід'ємними числами, які зазвичай знаходяться в діапазоні [0,1], 0 означає відсутність подібності, а 1 означає повну подібність.

Приклади міри подібності:

- Косинус-подібність п обчислюється шляхом знаходження косинуса кута між будь–якими двома взятими елементами. Формально, у матриці оцінок $n \times m$ (це матриця елементів користувача), подібність між будь-якими елементами дозволяє припустити, що ми беремо довільні елементи i та j, позначені як (2.11)

$$sim(i,j) = cos(i,j) = \frac{(i,j)}{\|i\|^2 * \|j\|^2} \tag{2.11}$$

- Зважена косинус-подібність – усуває недолік подібності на основі косинуса. Основна відмінність між оцінкою схожості в клієнтській спільній фільтрації та спільної фільтрації на основі елементів полягає в тому, що в умовах, коли CF на основі клієнта подібність обчислюється з урахуванням рядків матриці, де, як у випадку CF на основі елементів, спільність оцінюється, беручи разом стовпці. У той час як обчислення подібності з використанням косинусної міри базового рівня в розгляді на основі елементів, воно має одну суттєву ваду, а саме відмінності в оцінках, наданих користувачами між різними клієнтами, не беруться до уваги (2.12)

$$sim = \frac{\sum_{u \in U}(R(u,i) - R(u))(R(u,j) - R(u)}{\sqrt{\sum_{u \in U}(R(u,i) - R(u))^2}\sqrt{\sum_{u \in U}(R(u,j) - R(u))^2}} \tag{2.12}$$

- Подібність на основі кореляції – для цього показника порівнянності спільність між будь-якими двома елементами, припустимо, i та j, отримана шляхом реєстрації даної асоціації на основі подібності.

Припустимо, що набір клієнтів, які оцінили i та j, позначено U, тоді (2.13)

$$sim = \frac{\sum_{u \in U}(R(u,i) - R(u))(R(u,j) - R(u)}{\sqrt{\sum_{u \in U}(R(u,i) - R(u))^2}\sqrt{\sum_{u \in U}(R(u,j) - R(u))^2}} \quad (2.13)$$

- Розширений коефіцієнт Жаккара − ця міра подібності зменшується до коефіцієнта Жаккарда у випадку двійкових атрибутів. Розширений коефіцієнт Жаккара також відомий як коефіцієнт Танімото. Цей коефіцієнт, який представлено як EJ, визначається наступним рівнянням (2.14)

$$sim(i,j) = cos(i,j) = (x,y)\|x\|^2 + \|y\|^2 - (x,y) \quad (2.14)$$

Несхожість: це також числова міра ступеня відмінності об’єктів. Для більш схожих об'єктів несхожість є меншим значенням. Розбіжність падає в діапазоні [0,1], причому верхній діапазон змінюється від нуля до нескінченності. Приклади показників неподібності:

- Евклідова відстань.
- Відстань Мінковського.
- Манхеттенська відстань.
- Відстань Хеммінга.
- Подібність коефіцієнта Джеккарда [35].

Перейдемо до наступної метрики, яка буде розрахована для оцінки моделі, що вийшла, *nDCG*, або normalized discounted cumulative gain. Ця метрика часто використовується при оцінці пошукових систем і її суть полягає в оцінці результатів пошукової видачі на основі релевантності документів у пошуковій видачі та їх позиції в списку. Релевантність означає ступінь відповідності пошукового запиту та конкретного документа у видачі. Нормалізованість цієї метрики означає, що чим вище в списку релевантний документ, тим більше його значення впливатиме на підсумкове значення

метрики і навпаки відповідно. Можна зауважити, що в поставленому раніше завданні щодо створення рекомендаційної системи, з усіх її передбачень також важливі в основному лише перші фільми, які потенційно найбільше сподобаються користувачеві, якщо він їх подивиться. Тому *nDCG* метрику (2.15) можна застосувати для оцінки рекомендаційної системи. Релевантність у разі можна як прогнозовану оцінку фільму та її точність.

$$DCG_p = \sum_{i=1}^{p} \frac{r_i}{log_2(i+1)} = \sum_{i=1}^{p} \frac{2^{r_i} - 1}{log_2(i+1)} \quad (2.15)$$

де r_i – релевантність фільма на позиції i, а p – фінальна позиція [36].

2.6. Висновки до розділу

В результаті виконання даного розділу було обґрунтовано поняття рекомендаційних систем, які необхідні дані для її побудови, а також основні проблеми, які можуть виникнути під час її створення. Було визначено основні переваги та недоліки даних методів, а також проведено порівняльний аналіз між ними. І визначено, що найкращим варіантом є комбінація існуючих методів для уникнення проблем з «холодним запуском», розрідженістю даних, недостатньою новизною, тощо. Звичайно, що для кожного продукту має будуватись окрема рекомендаційна система і комбінація різних методів в залежності від заданих цілей. Саме тому було описано обраний метод та необхідні дані, які будуть збиратись з веб застосунку. Також було описано основні типи метрик, які необхідні для оцінки ефективності, а також їх основні переваги та недоліки.

3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1. Постановка задачі

На основі вхідних даних, що становлять історію прослуховувань користувача, зробити припущення про те, які композиції могли б його зацікавити у майбутньому та видати список рекомендацій, релевантні для слухача.

Для цього виконання поставленого завдання необхідно:

- Реалізувати базові алгоритми.
- Оцінити отримані результати за допомогою вибраних метрик.
- Застосувати гібридний підхід для комбінування базових підходів з метою покращення результатів.

3.2. Засоби розробки програмного забезпечення

HTML – це мова розмітки, яка визначає структуру вмісту. HTML складається з серії елементів, які використовуються для включення або обгортання різних частин вмісту, щоб він виглядав певним чином або діяв певним чином. Об'ємні теги можуть створювати гіперпосилання на слово чи зображення в іншому місці, виділяти слова курсивом, збільшувати або зменшувати шрифт тощо. HTML є основною мовою будь-якого веб-сайту. Він працює на різних платформах, включаючи CSS і JavaScript для ефективності та ефективності веб−сторінки [37]. Для прикладу візьмемо такий рядок вмісту:

JavaScript – це легка та відома мова сценаріїв. JavaScript можна описати як мову сценаріїв World Wide Web (W.W.W). Він поєднує різні перевірки веб-форм, функціональні можливості виявлення браузера та створення файлів cookie та ін. JavaScript є популярною та потужною мовою сценаріїв, яка підтримується більшістю веб-браузерів, таких як Opera, Firefox та Internet Explorer. Він завжди використовується в клієнтській веб-розробці. JavaScript також дозволяє веб-сторінкам бути динамічними та інтерактивними. Це

дозволяє безпосередньо імплантувати більшість мов сценаріїв, зокрема Java, у коди HTML [38].

Програмні інтерфейси браузера (API), вбудовані у веб–браузери, що забезпечують такі функції, як динамічне створення HTML і встановлення стилів CSS; збирання та маніпулювання відеопотоком із веб–камери користувача або генерування 3D–графіки та зразків аудіо.

Сторонні API, які дозволяють розробникам включати функціональність у сайти інших постачальників вмісту, наприклад Twitter або Facebook.

Фреймворки та бібліотеки сторонніх розробників, які можна застосувати до HTML, щоб прискорити роботу зі створення сайтів і програм.

Сценарій PHP – це програма з відкритим вихідним кодом, яка використовується для розробки та реалізації веб-сайтів. Це чудова мова сценаріїв, яка використовує кілька програмних технологій, як-от програмне забезпечення для онлайн-бізнесу для керування вмістом, інструменти розробки на динамічних веб-сайтах і програмне забезпечення для чатів. Він може створювати динамічні та інтерактивні веб-сайти, які можна вставляти безпосередньо в серце HTML-коду. Він сумісний із такими веб-серверами, як Microsoft IIS і Apache. PHP можна працювати в Інтернеті на сервері, оскільки він підтримує багато баз даних, таких як MySQL, Generic ODBC і Oracle. MySQL зазвичай використовується серед інших баз даних [39].

SQL – це Американський національний інститут стандартів (ANSI), який дозволяє отримати доступ до баз даних і маніпулювати ними. SQL може вставляти, отримувати та видаляти записи з бази даних, виконувати запити, оновлювати записи в базі даних, створювати нові таблиці баз даних і збережені процедури в базах даних, а також установлювати дозволи для процедур, таблиць і представлень.

Anaconda – це безкоштовний дистрибутив мов програмування Python із відкритим вихідним кодом для наукових обчислень (наука про дані, програми

машинного навчання, великомасштабна обробка даних, прогнозна аналітика тощо), мета якого – спростити систему керування пакетами та розгортання. Версіями пакетів керує система керування пакетами conda. Дистрибутив Anaconda містить пакети обробки даних, придатні для Windows, Linux і MacOS [40].

Для обчислень і аналізу нам потрібні певні бібліотеки Python, які використовуються для виконання аналітики. Потрібні такі пакети, як SKlearn, Numpy, pandas, Matplotlib, фреймворк Flask тощо.

SKlearn: він містить різноманітні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, посилення градієнта, k-середні та DBSCAN, і розроблений для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy.

NumPy: універсальний пакет для обробки масивів. Він надає високопродуктивний об'єкт багатовимірного масиву та інструменти для роботи з цими масивами. Це основний пакет для наукових обчислень на Python.

Pandas: одна з найпоширеніших бібліотек Python у науці про дані. Він забезпечує високопродуктивні, прості у використанні структури та інструменти аналізу даних. На відміну від бібліотеки NumPy, яка надає об'єкти для багатовимірних масивів, Pandas надає двовимірний об'єкт таблиці в пам'яті під назвою «Фрейм даних» [41].

Flask: – це легка структура веб-додатків WSGI. Він розроблений, щоб зробити початок роботи швидким і легким, з можливістю масштабування до складних програм. Це починалося як проста обгортка навколо Werkzeug.

PyTorch – це пакет Python, який надає дві функції високого рівня, тензорні обчислення (наприклад, NumPy) із сильним прискоренням графічного процесора, глибокі нейронні мережі, побудовані на системі автоградації на основі стрічки. Зазвичай PyTorch використовується або як

заміна NumPy для використання потужності графічних процесорів, або як дослідницька платформа глибокого навчання, яка забезпечує максимальну гнучкість і швидкість. PyTorch має унікальний спосіб побудови нейронних мереж: використання та відтворення рекодера.

3.3. Функціональні можливості веб застосунку

Перед реалізацією рекомендаційної системи визначимось з функціоналом та набором функціональних можливостей, які небхідні будуть в різних сценаріх використання данної системи. Інтерфейс будет реалізований у вигляді веб застосунку, оскільки зазвичай користувачі використовують телевізор, комп'ютер чи ноутбук для перегляду фільмів чи серіалів, а не мобільний пристрій.

1. Для отримання доступу до системи користувач має авторизуватись:
    - Вхід: ввести логін та пароль
    - Реєстрація: заповнити форму, вказавши персональні дані:
        - Логін
        - Пароль
        - E-mail

    Після цього при першому вході необхідно заповнити демографічну інформацію:
    - країна користувача;
    - мова користувача;
    - соціальний статус користувача;
    - вік користувача.
2. Коли користувач отримав доступ до системи йому доступна головна сторінка з персоналізованою сторінку з контентом. Рекомендації повинні базуватися на вподобаннях користувача.

3. Після того, як користувач обрав фільм, він може додати його до списку вподобань та поставити оцінку і рекомендації мають перерахуватись, враховуючи оновлену інформацію.

Тож формалізуємо та виділимо основний функціонал та можливості системи:

- Надавати список рекомендованих фільмів користувачу.
- Авторизація та аутентифікація користувача в системі.
- Переглядати картку фільму з описом та рейтингом.
- Враховуються раніше переглянутих фільмів в рекомендаціях.

Система складається з трьох, так званих, модулів:

- модуль рекомендацій;
- серверна частина;
- веб застосунок;

3.4. Вимоги до розробленого програмного забезпечення

В результаті аналізу наявних програмних рішень, а також потреб для створеного методу було створено певні вимоги до кожного модуля:

Модуль рекомендацій:

- Модуль повинен мати доступ до бази даних історії переглянутих фільмів та карточок фільмів.
- Рекомендації фільмів повинні базуватися на вподобаннях користувача.
- Обчислення рекомендацій повинно займати не більше 5 секунд.
- Використання REST API для взаємодій з рекомендаційним модулем.
- Модуль повинен мати можливість легко горизонтально масштабуватися.

Серверна частина:

- Взаємодія з рекомендаційним модулем.
- Збереження інформації про користувача та його історії.
- Надання списку рекомендаційних фільмів, обробка операції покупки товару.
- Ідентифікація, авторизація та аутентифікація користувача.

Веб застосунок:

- Можливість взаємодії з REST API серверної частини.
- Відображення списку рекомендованих фільмів, а також фільмів з каталогу.
- Надання форми для входу користувача.

3.5. Архітектура веб застосунку

Після того, як основні вимоги були визначенні, переходимо до побудуви архітектуру системи. В архітектурі рекомендаційної системи потрібно враховувати масштабованість системи. В сучасних подібних архітектурах використовуються хмарні обчислювальні платформи так, як вони можуть забезпечити автоматичне горизонтальне масштабування системи.

При побудові веб застосунку будемо використовувати трирівневу клієт-сервісну архітектуру (рис. 3.1). Інтерфейс користувача представляє собою веб-сторінку, з якою взаємодіє користувач застосунку. База даних містить персональну інформацію про користувача, а також про його взаємодію з застосункою. Сервер – проміжний рівень, яки використовується для взаємодії рівня інтерфейсу з базою даних. Сервіс дозволяє розміщувати дані клієнти, використовувати їх для обробки. За допомогою серверу відбуваються обчислення в алгоритмах, будується нейронна мережа. Ткож сервер виконує процес авторизації та аутентифікації користувачів.

Частина інтерфейсу взаємодіє з сервером по мережі і демонструє результати виконання обчислень серверу. Така архітектура підвищує

ефективність і продуктивність системи через правильне розподілення задач між рівнями. Особливо це важливо в умова навантаженості, коли необхідно обробити велику кількість даних або виконати складні обчислення.

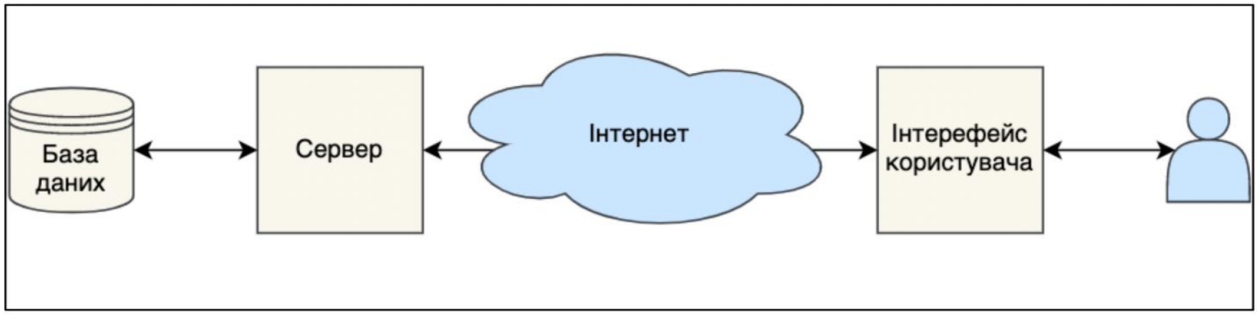


Рис. 3.1. Трирівнева клієнт-сервісна архітектура

3.6. Вимоги до технічного забезпечення

Програмне забезпечення, яке використовувалось при розробці даної рекомендаційної системи:

- Платформа Windows.
- Мова Python, версія 3+.
- Фреймворк Angular, версія 10.0.0+.
- Мова NodeJS, версія 14.15.4.
- TypeScript, версія 3.1+.
- MySQL, версія 5.7+.
- PHP 5.2+.
- Доступ до швидкої мережі Інтернет.

Загальні вимогами до технічного забезпечення клієнтського застосування:

- веб браузер Chrome, Safari, Ethernet Explorer, Firefox (підтримуються останні версії браузерів);
- доступ до мережі Інтернет.

3.7. Створення моделі машинного навчання

Машинне навчання в роботі використовувалось для швидкого пошуку рекомендованого фільму, реалізовано за допомогою бібліотек мови Python. Графічне зображення процесу створення моделі машинного навчання з учителем (рис. 3.2).

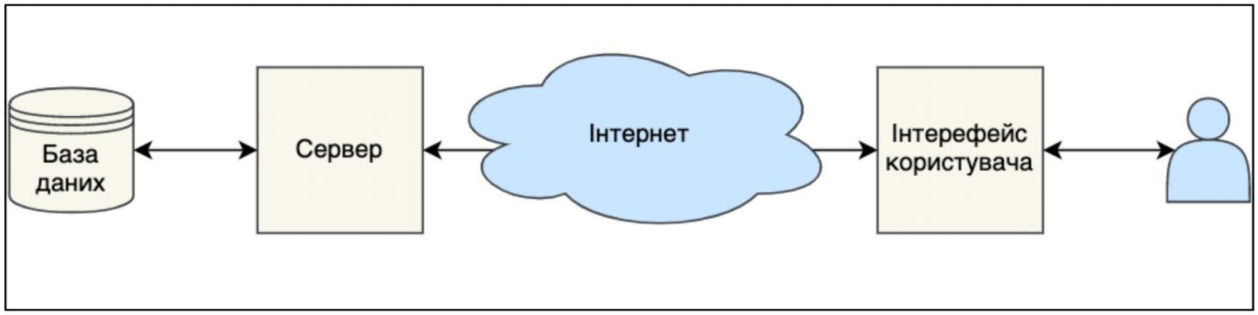


Рис. 3.2. Процес створення моделі машинного навчання

Піля того, як модель завершить навчатись на тестових даних, вона вже зможе самостійно навати рекомендації користувачам веб-застосунку.

Після завершення підготовки навчального набору, наступним етапом буда розробка та реалізація нейронної мережі (рис. 3.4) віповідної архітектури, у ході якого були самостійно визначені такі її параметри, як кількість рівнів та розмірність їх виходів.

В отриманій моделі роль вхідного шару виконують два незалежні тензори для фільму та користувача відповідно. Потім, one-hot вектор, що означає унікальний номер користувача, пов'язаний з рівнем векторного представлення. Далі нейронна мережа містить об'єднуючий рівень, один із входів якого це вже передопрацьована інформація про фільм, а другий − отриманий власний вектор користувача. Після цього в моделі слідують 5 повнозв'язкових прихованих рівнів з функціями активації у вигляді ReLU. Наприкінці мережі вихідний рівень з функцією сигмоїдальної активації видає

відповідь у вигляді ймовірності взаємодії користувача і фільму, на основі якої і здійснюється рекомендація.

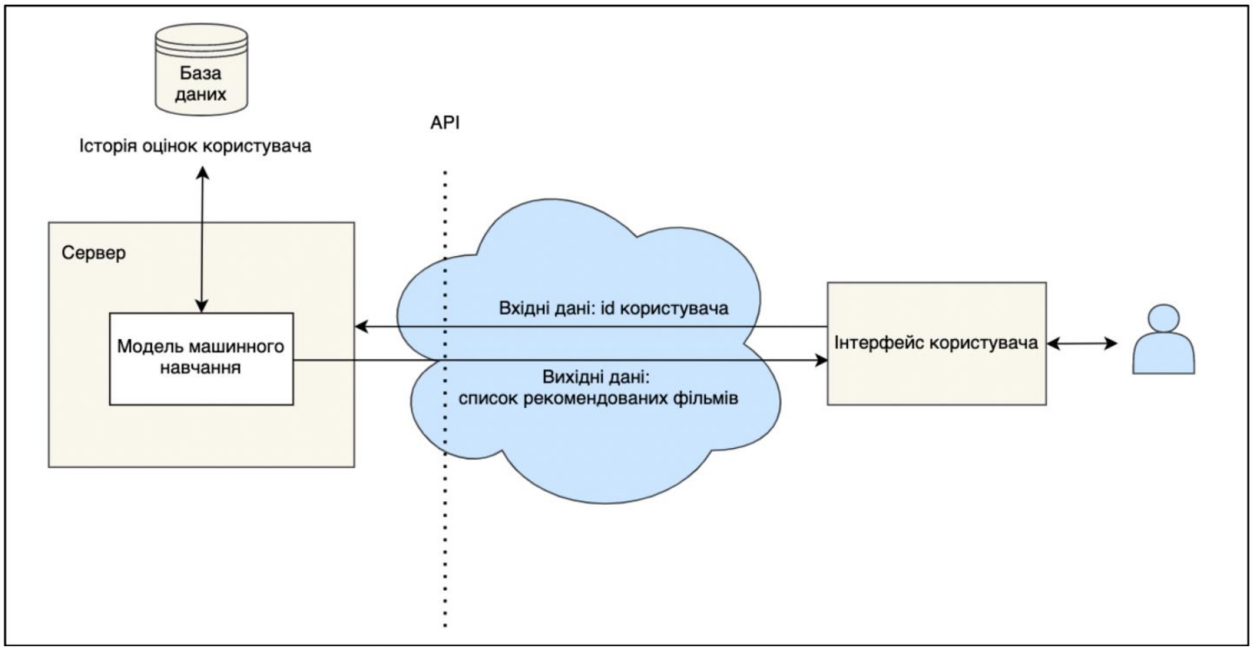


Рис. 3.3. Схема використання моделі машинного навчання для надання рекомендацій кінцевим користувачам

3.8. Висновки до розділу

Під час написання роботи було сформовано постановку задачі, обрано програмних забезпечень для реалізації веб застосунку, було детально описано функціональні можливості та вимоги до веб застосунку та технічного забезпечення. Було опсано вискорівненву архітектуру програмного забезпечення і нейронної мережі, а також моделі машинного навчання.

4. РЕАЛІЗАЦІЯ ВЕБ ЗАСТОСУНКУ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

4.1. Вхідні дані: обробка і аналіз

Як база даних для рекомендаційної системи був обраний найпопулярніший і найширший набір даних “MovieLens”. Цей датасет реальну інформацію і був спеціально зібраний дослідницькою лабораторією для рекомендаційних мереж, їх вивчення, аналізу, розробки та експериментів. “MovieLens” містить дані про 283228 користувачів та їх оцінки яких в сумі 27753444. Наявні оцінки відносяться до 58098 фільмів, про які є короткі відомості: назви та список жанрів. Оцінки можуть приймати значення від 1 до 5 з кроком 0,5.

Основним файлом даного датасету є “ratings.csv”. Він складається із 4 колонок: у перших двох колонках записані унікальні ідентифікатори користувача та фільму, у третій колонці міститься оцінка, яку цей користувач присвоїв вказаному фільму, а в четвертій колонці − ідентифікатор часу цієї події (рис 4.1).

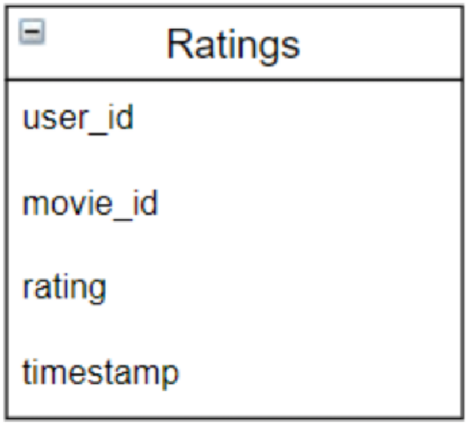


Рис. 4.1. Структура таблиці з рейтингами

Оскільки “MovieLens” містить невелику кількість інформації про фільми, вона була доповнена за допомогою “The Movies Dataset”. В цьому

наборі міститься інформація про 45 000 фільмів, випущених до липня 2017 року включно. Датасет складається з файлів формату .csv, які включають всю доступну інформацію про фільми. Основні дані містяться у файлі “movies.csv”. Там записані назви, описи, жанри, країни виробництва, бюджети, касові збори, дати виходу в прокат та багато іншого для кожного з фільмів. Однак, їх ключові слова винесені до окремого файлу “keywords.csv”. Також, ціла група інформації про фільми міститься в іншому файлі “credits.csv”, а саме акторський склад, режисер та інші члени команди тощо (рис. 4.2).

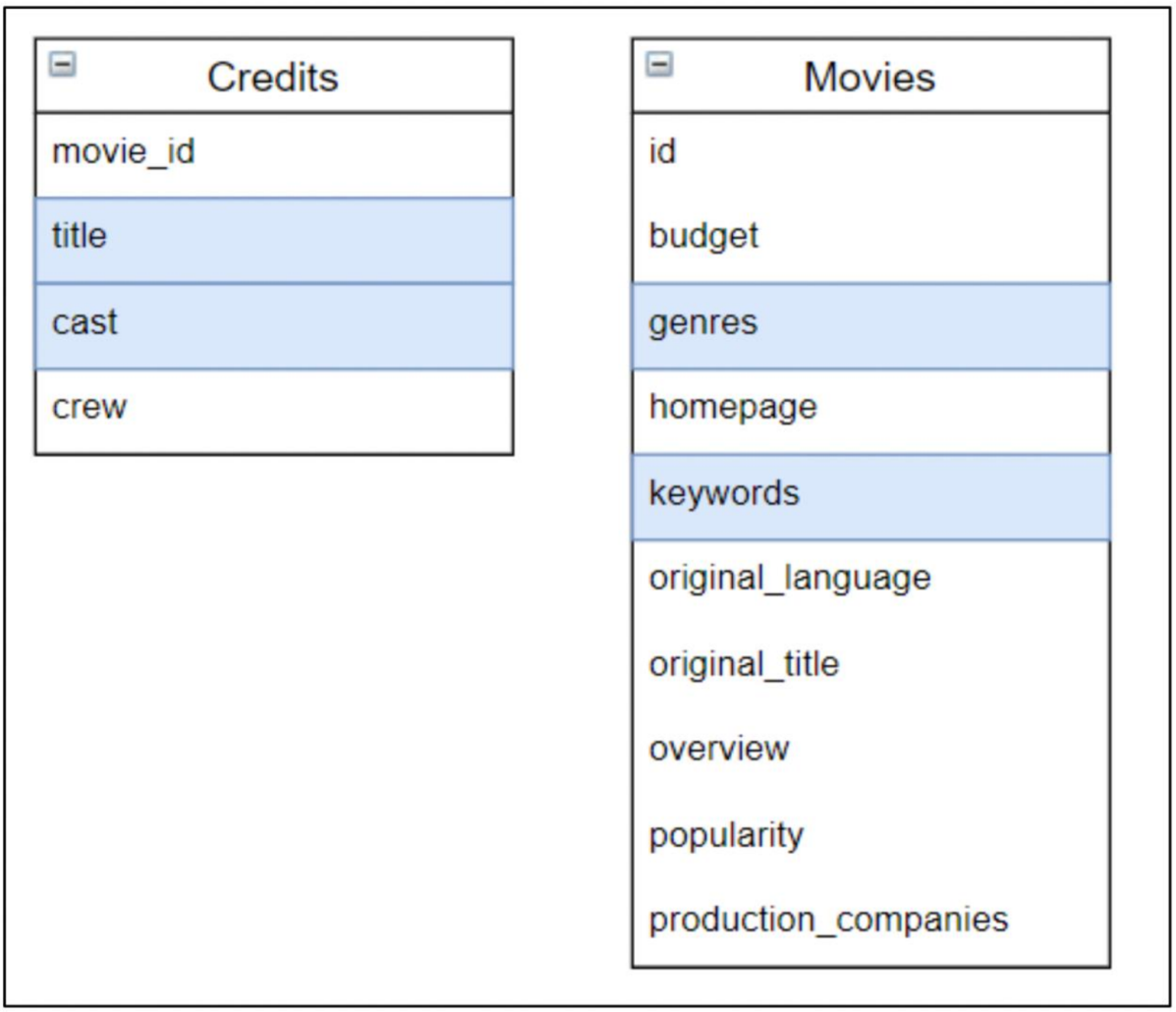


Рис. 4.2. Модель метаданих про фільми

4.2. Прототип інтерфейсу веб застосунку

В веб застсунку на кожнійсторінці вверху є меню з навігацією. Відкривши розроблений веб застосунок користувач потрапляє на Головну сторінку (рис. 4.3), вона вже показує список рекомендацій для користувача. Якщо користувач не авторизувався, то він буде бачити список рекомендацій в порядку популярності фільмів серед усіх користувачів веб застосунку.

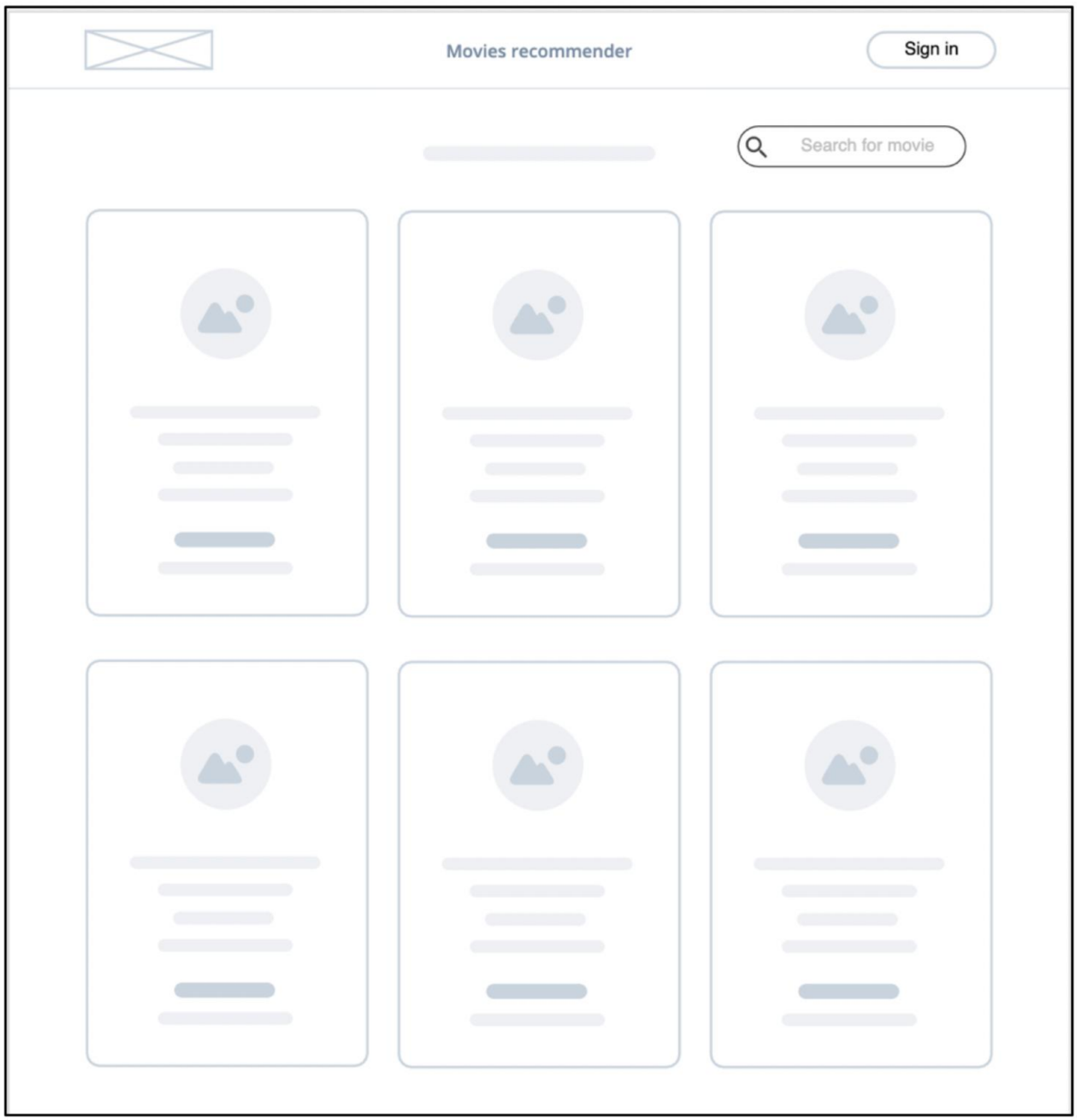


Рис. 4.3. Прототип інтерфейсу головної сторінки

Далі користувач може авторизуватись ввівши свій e-mail та пароль (рис. 4.4). На сторінці знаходіться форма для входу в систему, що містить поля «Email», «Password». Заповнення цих полів обов'язкове для входу в систему. Валідація значень полів форми входу не передбачена. Після перенаправляється на «Головну сторінку», якщо дані при реєстрації вказані некоректно, на екран виводиться повідомлення про помилку «Username or password is incorrect». Використання такого підходу до валідаціїї зменшує ризики злому пароля, а отже підвищує рівень безпеки користувача під час використання додатку.

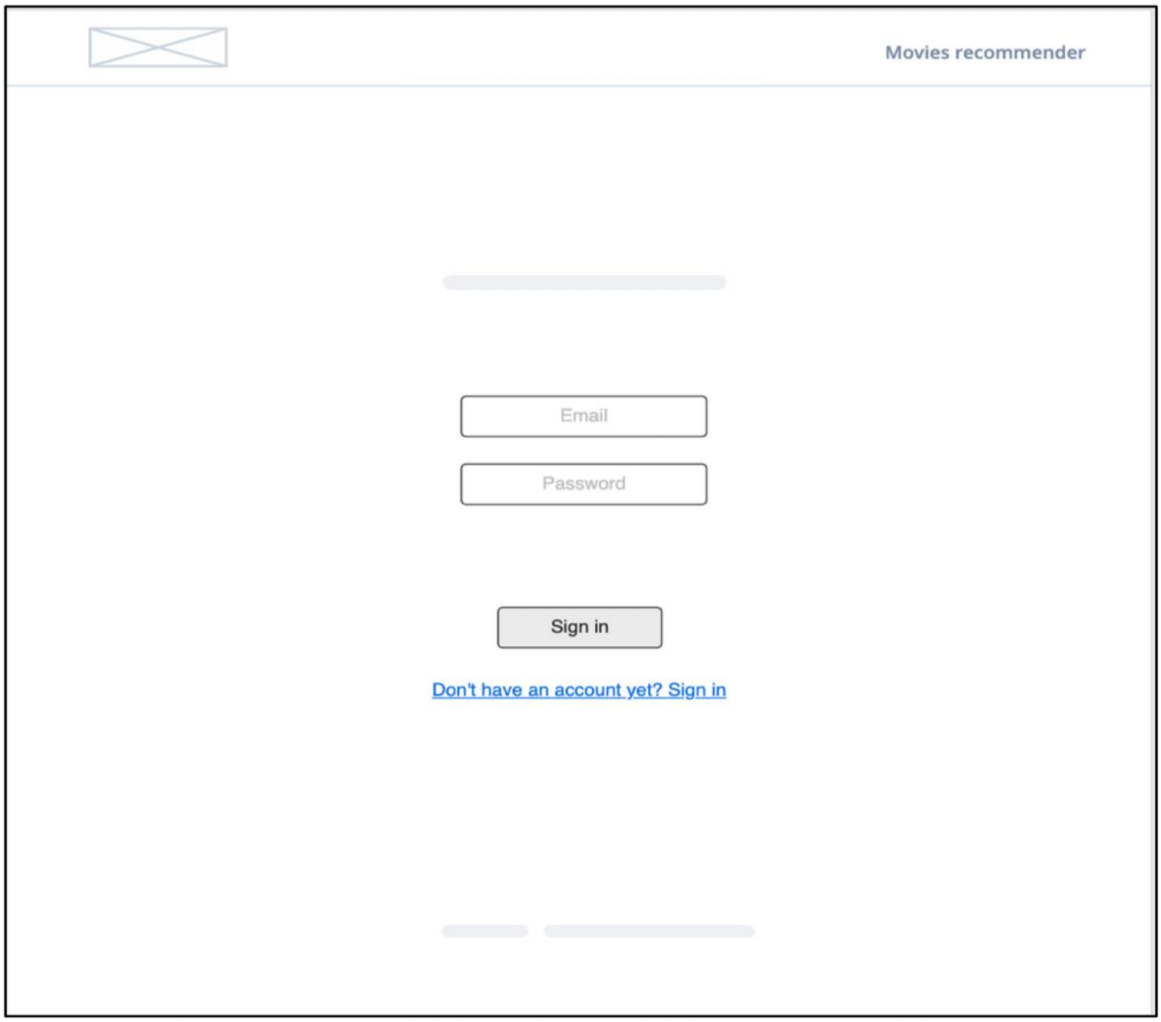


Рис. 4.4. Прототип сторінки входу в систему

Якщо у користувача не має облікового запису він може зареєструватись, перейшовши на сторінку реєстрації. На сторінці пропонується форма реєстрації, що складається з полів «Legal name», «Email», «Password», «Repeat password» (рис. 4.5). Далі користувачу пропонується заповнити форму с демографічними данними, для використання методом демографічної фільтрації.

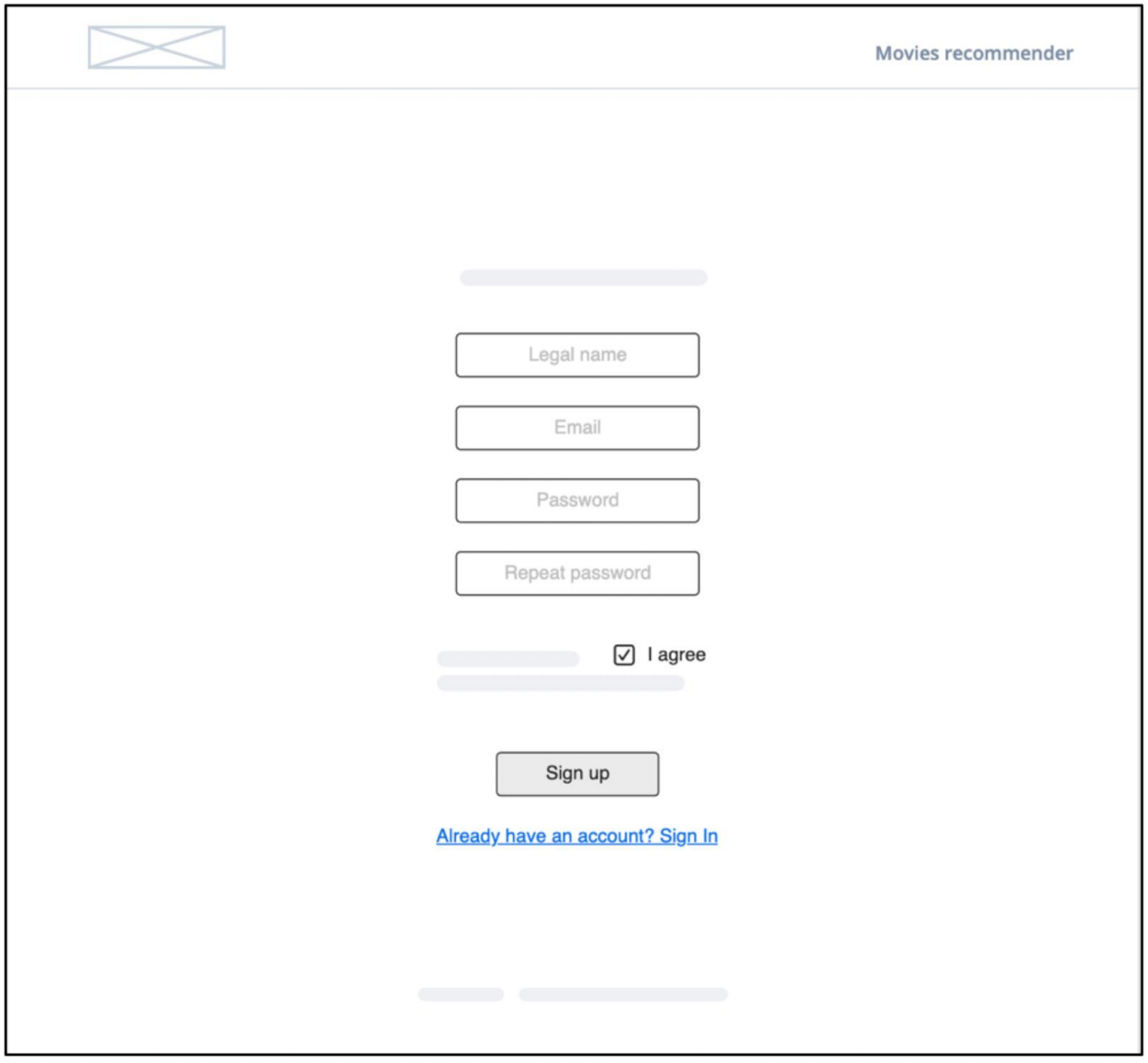


Рис. 4.5. Прототип сторінки реєстрації

Необхідно правильно заповнити всі поля, тому що це обов'язкова умова для продовження процедури реєстрації. Поля форми реєстрації підпорядковуються таким правилами:

- legal name – обов'язкове поле, містить мінімум два символи.
- email – обов'язкове поле, містить строку email формату.
- password – обов'язкове поле, містить від 8 до 16 символів, мінімум одну літеру, мінімум одну цифру.
- repeat password – обов'язкове поле, дорівнює значенню password поля.

Коли користувач натискає на кнопку «Sign up» створюється обліковий запис користувача в системі і відкривається «Сторінка входу в систему». Якщо хоча б одне поле не було заповнено, або заповнено помилково, на екран виводиться валідаційна помилка, некоректне поле підсвічується червоним кольором, кнопка «Sign up» недоступна. В такому випадку процес реєстрації може бути продовжено лише коли користувач змінить дані в некоректному полі.

Якщо натиснути на будь-який фільм зі списку фільмів, з'явиться сторінка з інформацією про фільм. (рис. 4.6) Авторизованому користувачу доступна функція веб застосунку – можливість оцінити фільм.

Для оцінки натискаються кнопки у вигляді зірочки. Якщо ви ще не оцінили фільм або не ввійшли в систему, рейтинг на сторінці інформації – це середня оцінка фільму глядачами плюс округлення до цілого числа від одного до п'яти. Якщо фільм вже був оцінений поточним користувачем, то оцінка з'явиться як при натисканні з можливістю її зміни. Як правило, не авторизований користувач, не має доступу до кнопки у вигляді зірооик, як і взагалі функція оцінки фільму.

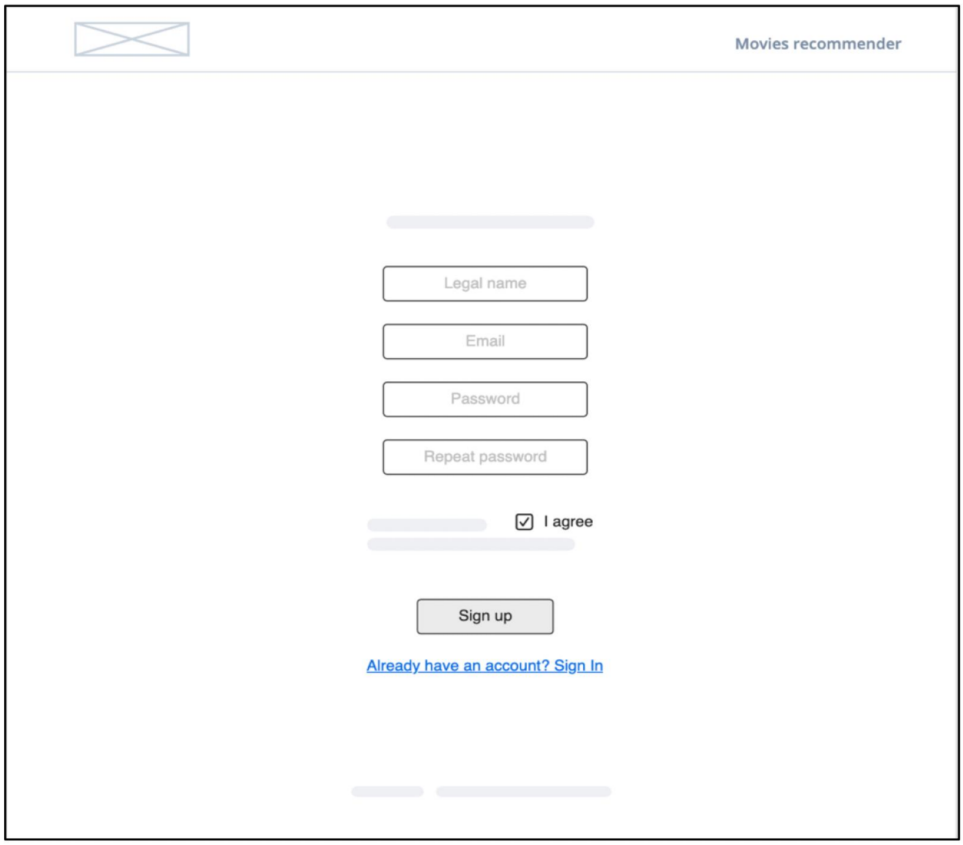


Рис. 4.6. Прототип сторінки деталей про фільм

4.3. Опис розробленого програмного забезпечення та його використання

Інтерфейс веб-застосунку було розроблено на основі прототипу з використанням обраних технологій. Для неавторизованого користувача на головній сторінці відображається список фільмів, рекомендованих на основі рейтингу серед усієї аудиторії додатку (рис. 4.7).

Тільки після входу в систему можна отримати персоналізований контент на головній сторінці. Ввійти в систему можна лише після заповнення форми входу (рис. 4.8).

Після входу в систему клієнтська частина додатку отримує від сервера JSON Web Token, завдяки якому користувач може взаємодіяти з особистими

даними в системі. JSON Web Token – це відкритий галузевий стандарт (RFC 7519) для надійного взаємозв'язку між двома сторонами. На клієнтській частині веб-додатку JWT зберігається в глобальному Redux сховищі.

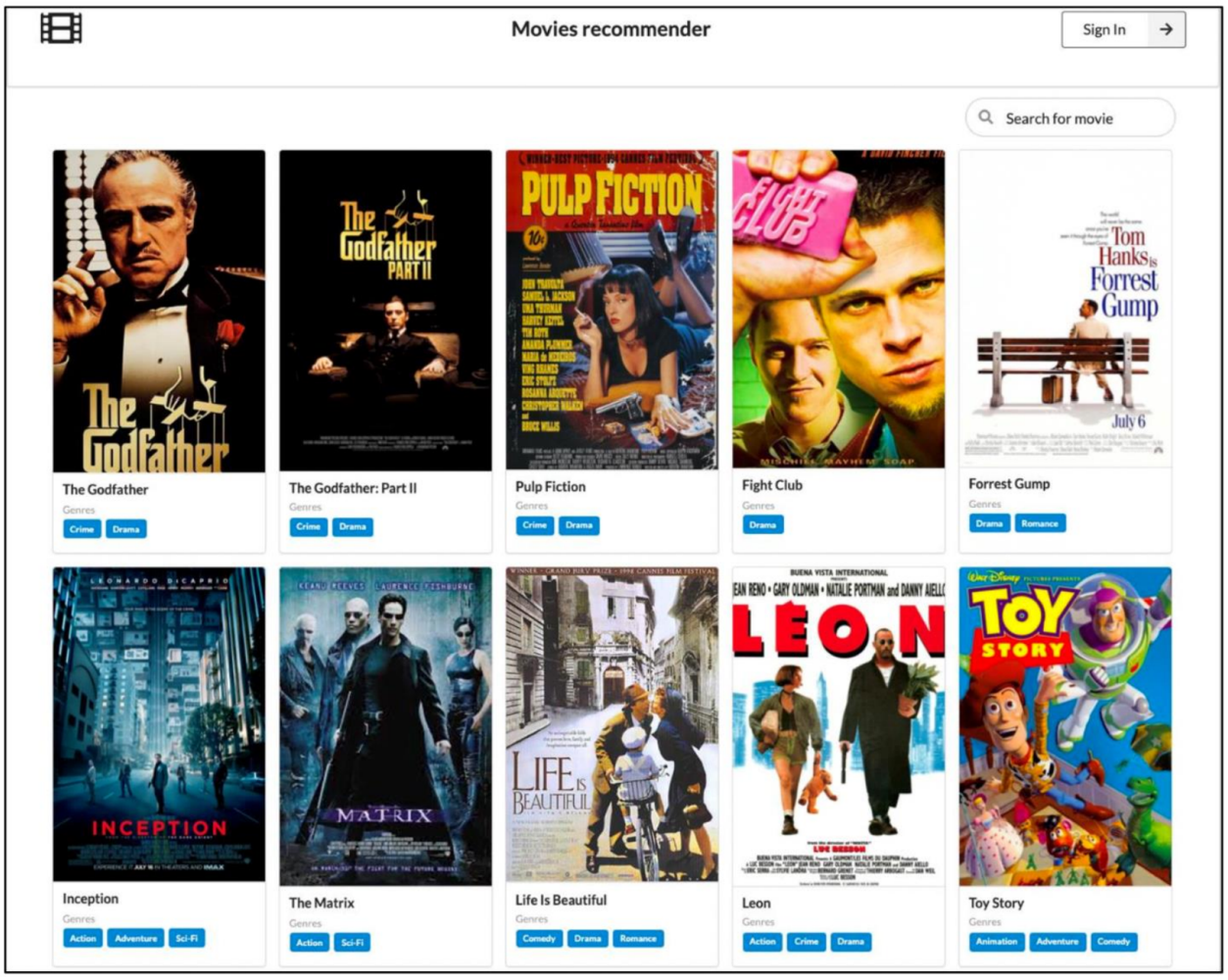


Рис. 4.7. Головна сторінка для неавторизованого користувача

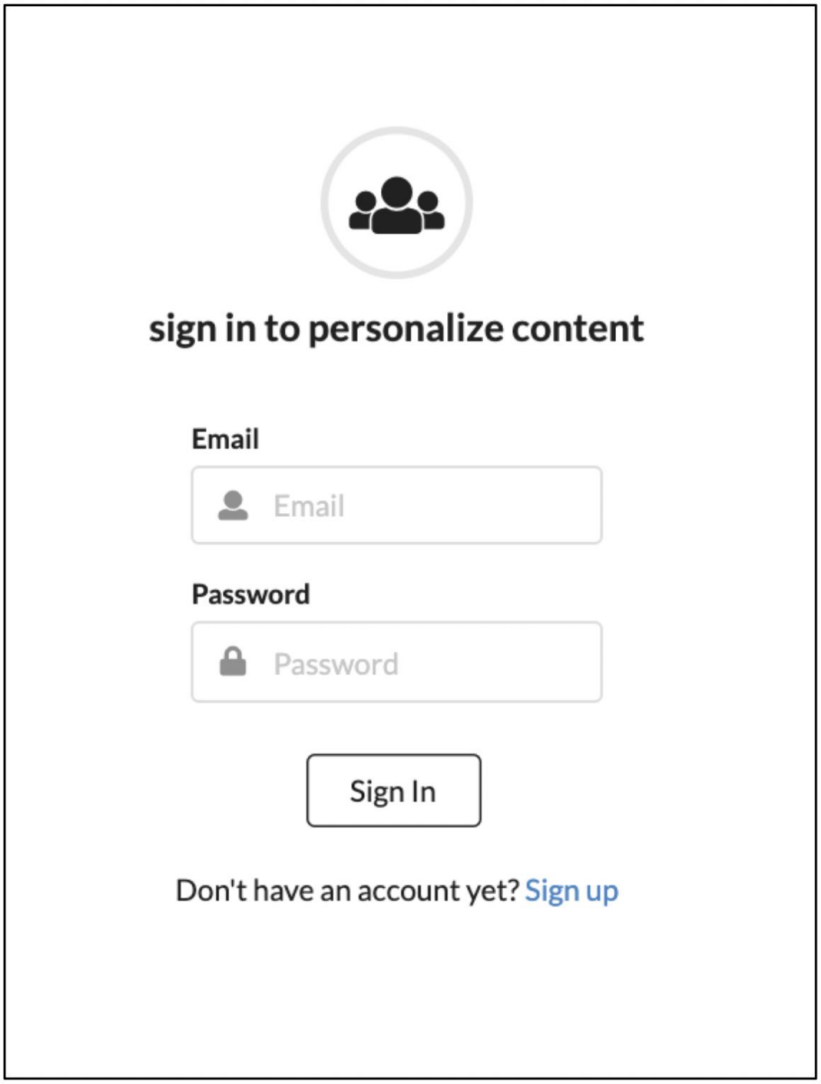


Рис. 4.8. Логін форма

Для нових користувачів необхідно створити обліковий запис через форму реєстрації (рис. 4.9) і після завершення реєстрації повернутися на сторінку входу.

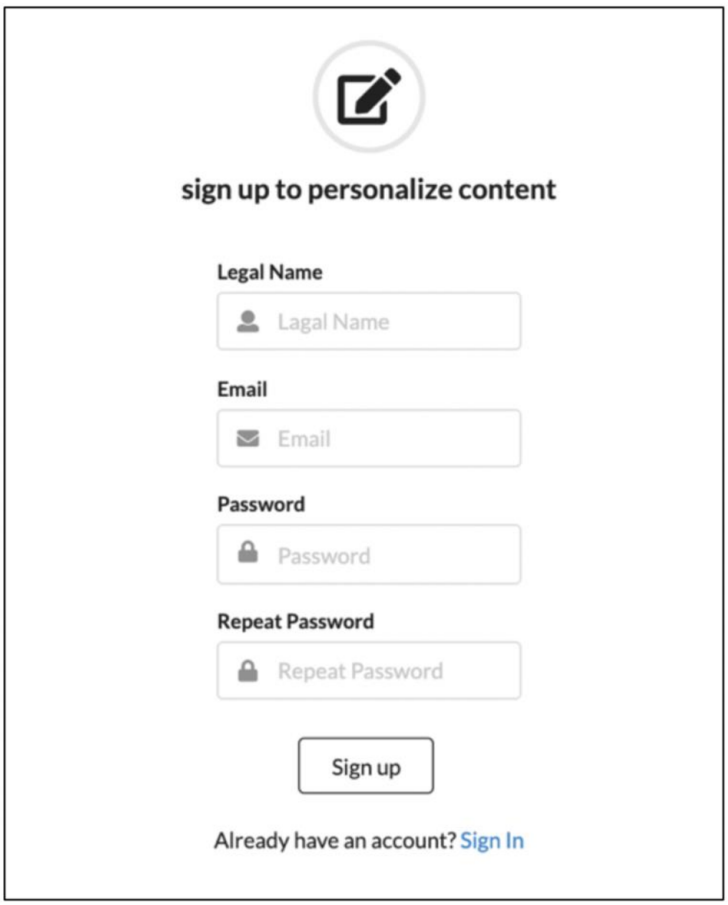


Рис. 4.9. Форма реєстрації

На сторінці інформації про фільм (рис. 4.10) можна оцінити запропонований контент за шкалою від одного до п’яти. Щоб перейти на цю сторінку, достатньо просто натиснути на картку вибраного фільму на головній сторінці програми. Щоб змінити вже надісланий рейтинг, також можна перейти на сторінку деталей, натиснувши на рядок із відповідним фільмом на сторінці історії рейтингів.

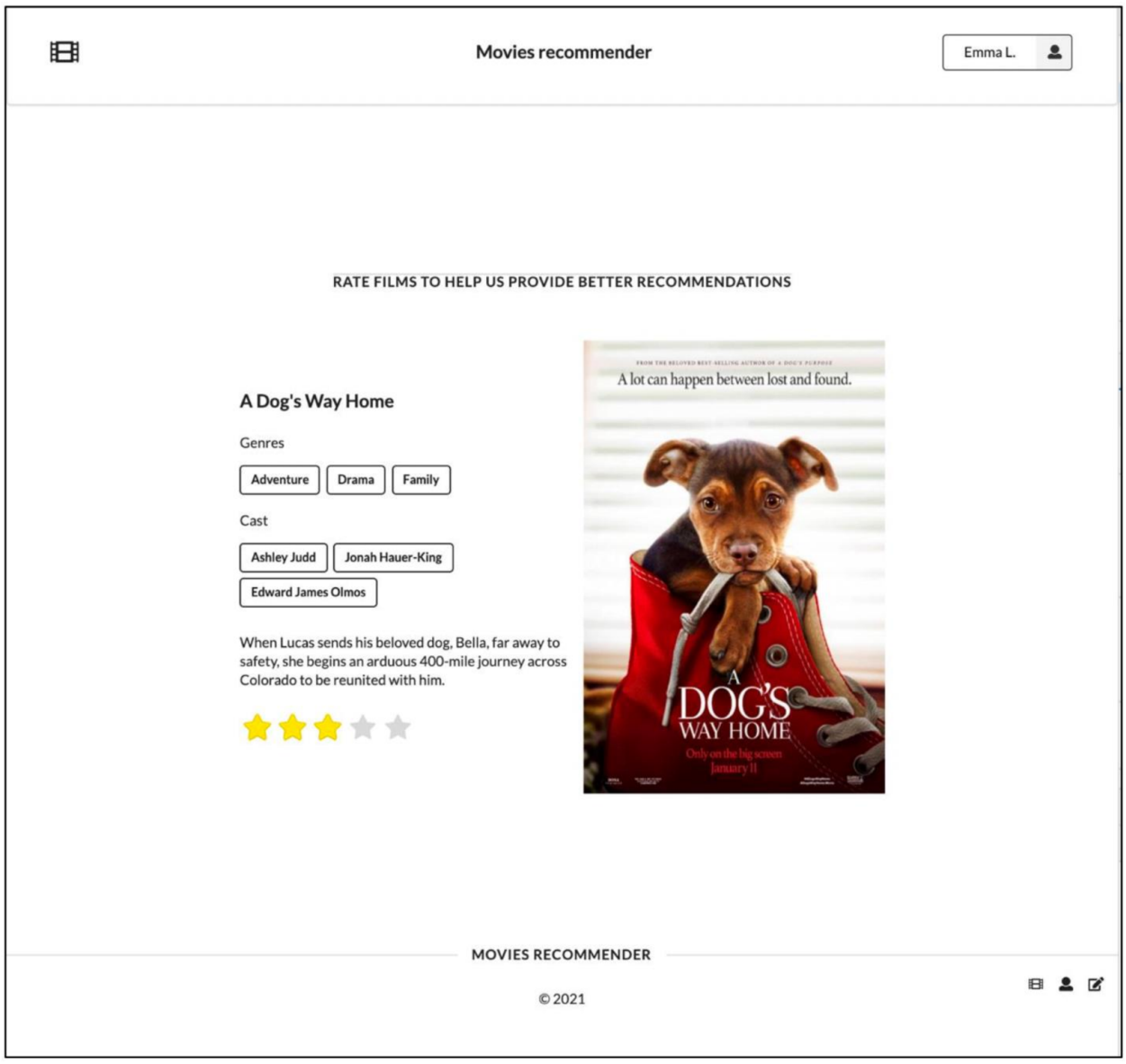


Рис. 4.10. Сторінка деталей фільму

Можна оцінити фільм за допомогою декількох кнопок у формі зірки. Ця функція програми доступна лише авторизованим користувачам. За замовчуванням на сторінці відображається кількість зірок, що відповідає середній оцінці фільму аудиторією програми.

Для авторизованого користувача, який уже оцінив принаймні один фільм, на головній сторінці відображається персоналізований список рекомендованих фільмів для перегляду. Він створюється на основі

розробленої та підготованої на тренувальних даних моделі машинного навчання. Розглянемо рекомендації, надані системою конкретному користувачеві (рис. 4.10) та список оцінених ним фільмів (рис. 4.11).

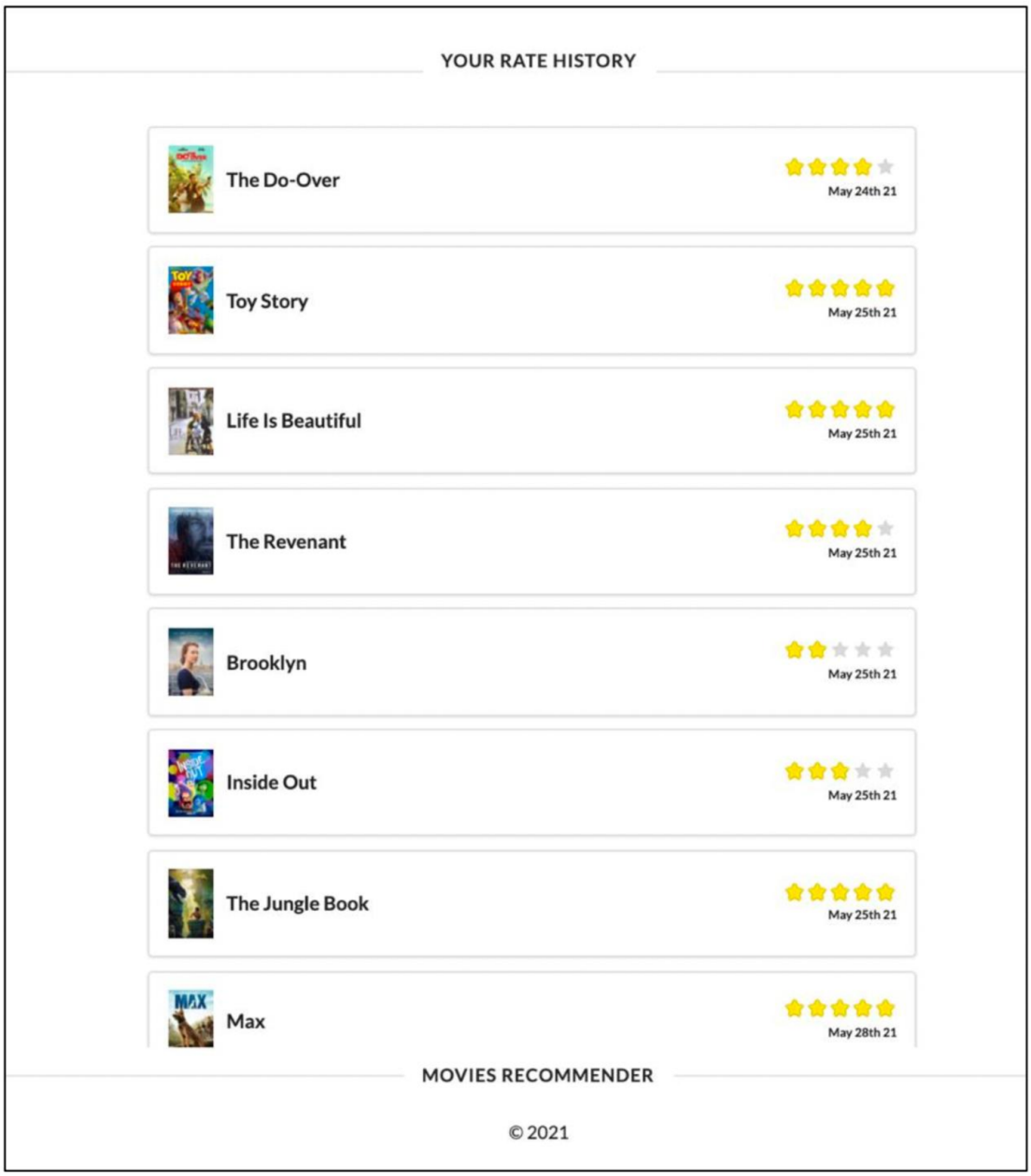


Рис. 4.11. Історія оцінок користувача

Сторінка історії рейтингів (рис. 4.11) показує, що відповідний користувач дав високі оцінки комедіям, сімейним та анімаційним фільмам.

Користувача можна віднести до типу простого оцінювача, оскільки майже всі його оцінки вище середнього значення.

Бачимо, що рекомендований контент (рис.4.12) схожий на той, що має високі рейтинги від користувача, однак не дублює його і не є одноманітним. Такі рекомендації, сприяють знаходженню нового цікавого та актуального контенту і розширенню інтересів користувача.

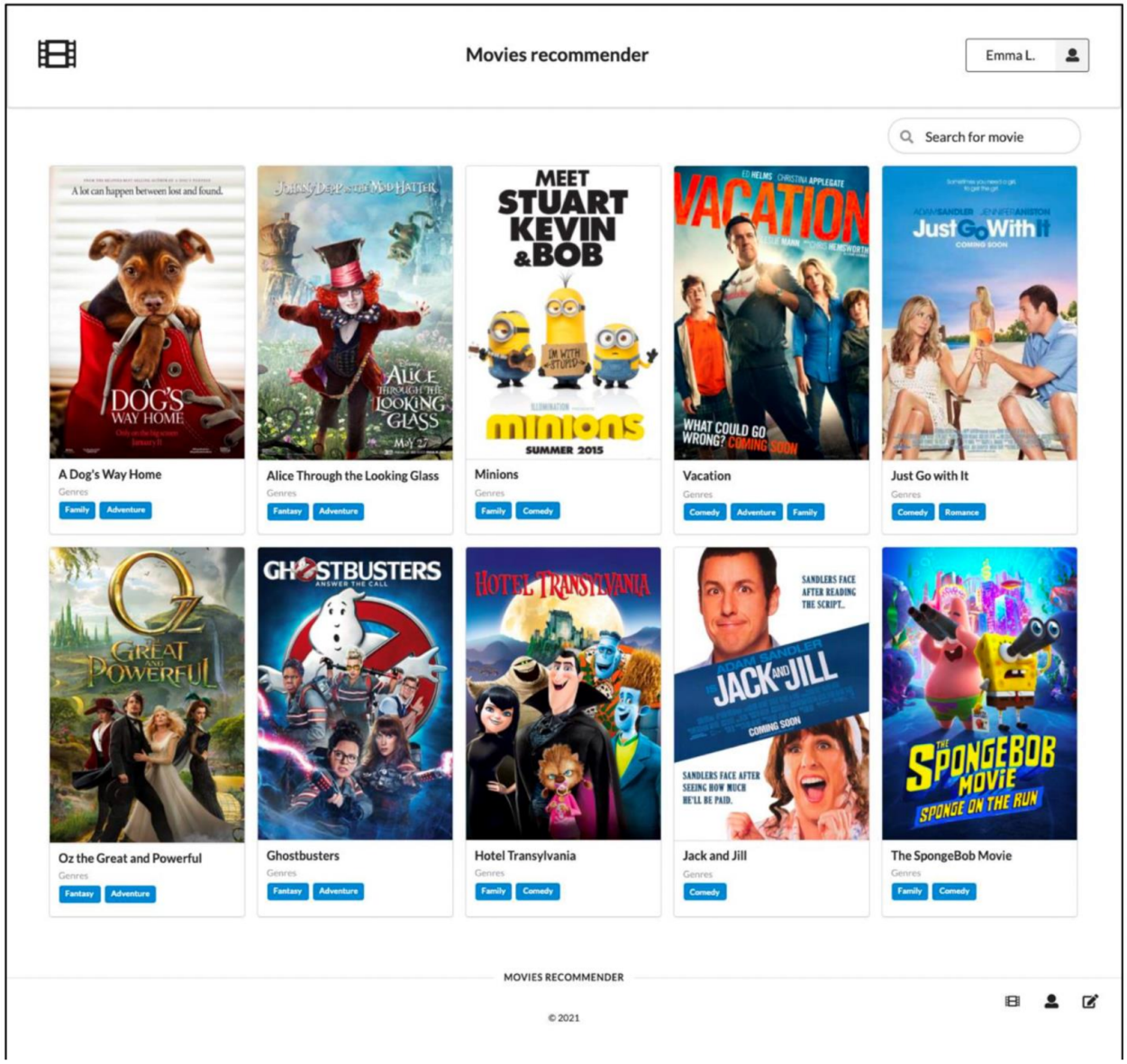


Рис. 4.12. Головна сторінка для авторизованого користувача

4.4. Експериментальні дослідження з використанням розробленого програмного забезпечення

Для початку побудуємо графік на основі входження кожного жанру у фрейм фільмів (рис. 4.13).

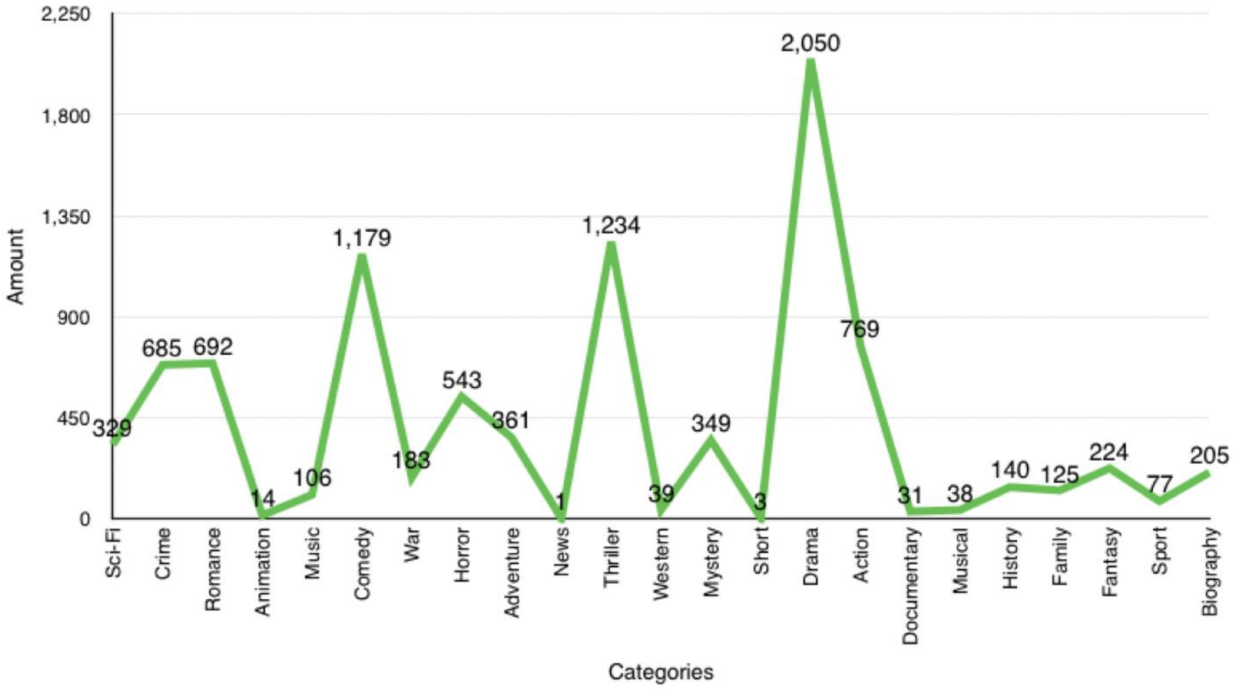


Рис. 4.13. Графік кількості входжень у фрейм фільмів

Як бачимо в жанрі Драма більше всього фільмв, а в жанрі новин найменше – всього 1.

Для кожного жанру підрахуємо F-measure гібридного підходу (зелений) і метод колаборативної фільтрації (блакитний) (рис. 4.14).

Як бачимо по всім жанрам гібридний підхід найбільший.

Для перевірки ефективності розробленої моделі машинного навчання для розв'язку задачі персоналізації необхідно перевірити точність рекомендацій які вона надає.

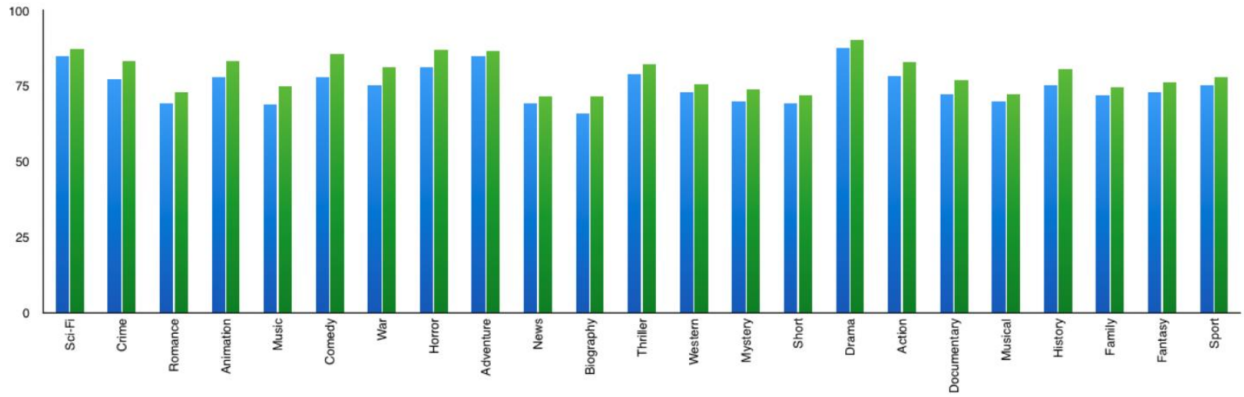


Рис. 4.14. Порівняння F-measure для двох методів по жанрам

Експерименти будемо проводити використовуючи різні комбінації вищезазначених алгоритмів:

- колаборативна фільтрація (CF);
- фільтрація на основі вмісту (CB);
- демографічна фільтрація (DF);
- CB + CF;
- CF + DF;
- DF + CB;
- модифікований підхід (CF+CB+DF).

Результати експериментів зображені на рис. 4.15.

Як видно з результатів експериментів – наш підхід дає досить точні прогнози, в порівнянні з іншими підходами (точність – 0.722, повнота – 0.7, F-міра – 0.71).

Тепер перейдемо до перевірки результатів навчання нейронної мережі. В ході навчання критерії якості вимірялися двома способами. Раз в 10000 кроків виміряно якість на всьому тестовому виборі, раз в 100 кроків – на одному випадково вибраному значенні користувача. В ході навчання критерії якості вимірялися двома способами. Раз в 10000 кроків виміряно якість на всьому тестовому виборі, раз в 100 кроків – на одному випадково вибраному

значенні користувача (рис. 4.16 і рис. 4.17)

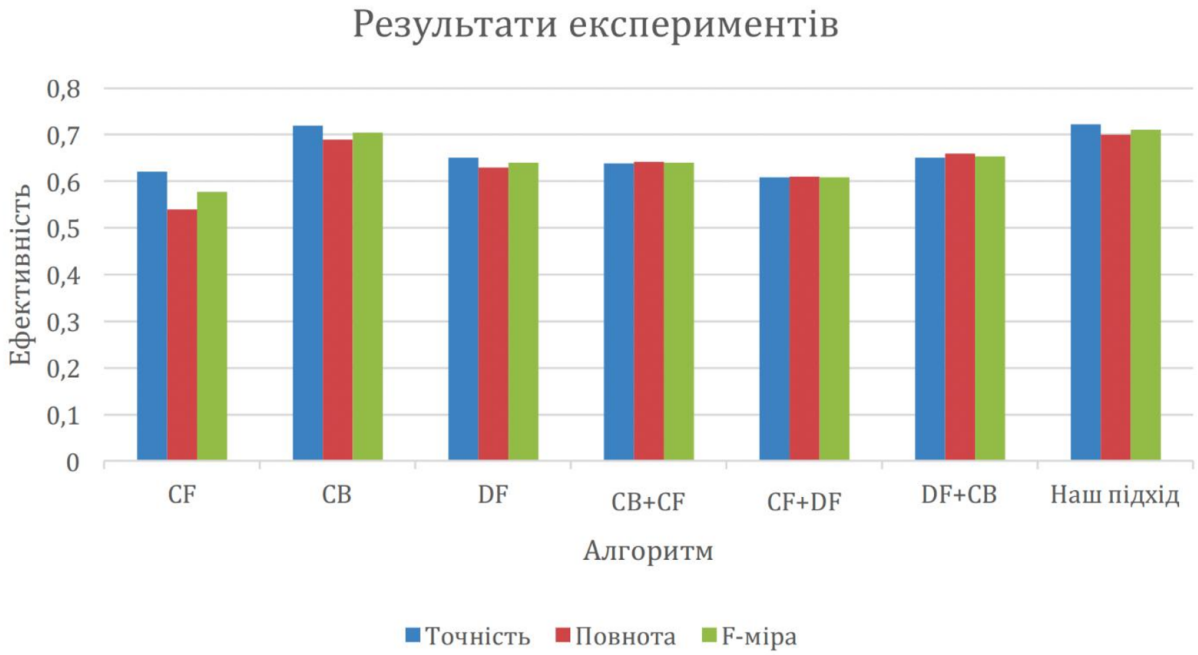


Рис. 4.15. Порівняння підходів по метрикам

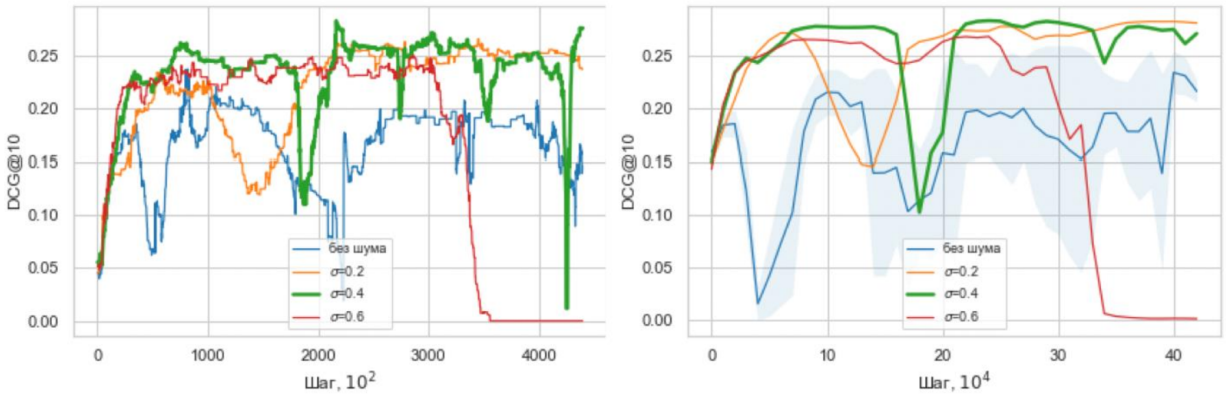


Рис. 4.16. Значення DCG раз в 100 і 10000 кроків на тестовому наборі

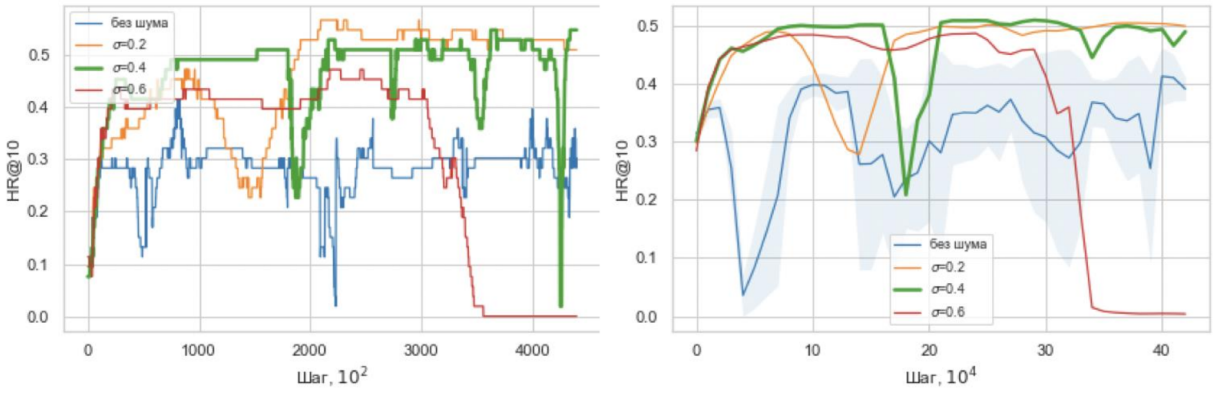


Рис. 4.17. Значення HR раз в 100 і 10000 кроків на тестовому наборі

Для підсумкового обчислення якості моделі використовувалися найкращі ваги з історії оцінювання всім користувачам. Результати представлені у таблиці 4.1.

Таблиця 4.1

Порівняння різних варіантів шуму

Модель	DCG@10	HR@10
σ = 0.6	0,268	0,487
σ = 0.4	0,282	0,509
σ = 0.2	0,282	0,504
Без шуму	0,254	0,454
Випадкові рекомендації	0,05	0,1

Видно, що використання шуму підвищує як підсумкові критерії якості, і їх проміжні значення у процесі навчання.

Для навчання було обрано оптимізатор “Adam”, воно проходило у 100

епох, на кожному кроці розглядалося 64 навчальні приклади. В результаті тренування даної нейронної мережі було отримано такі результати:

- значення функції помилки − 1.3 (рис. 4.18)

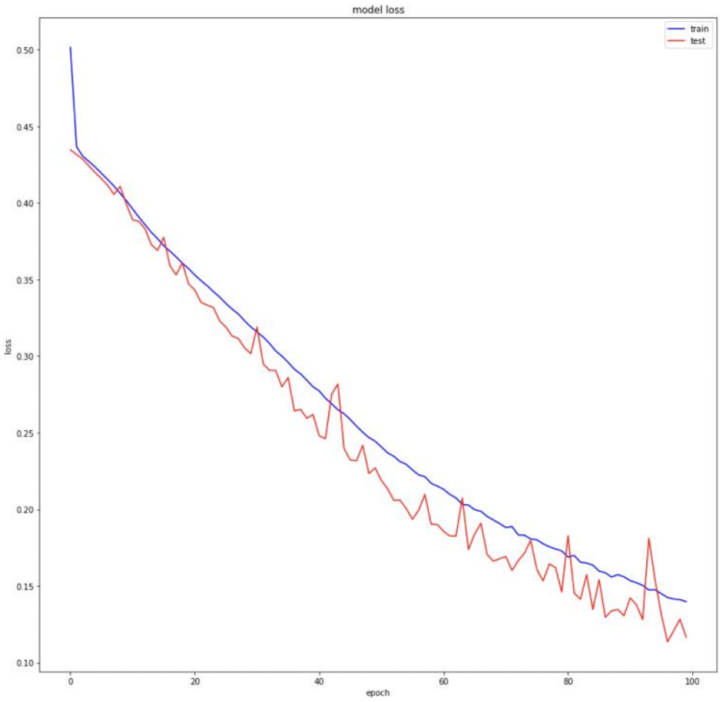


Рис. 4.18. Графік функкції помилки

- значення метрики “Accuracy” = 0.95 (рис. 4.19)

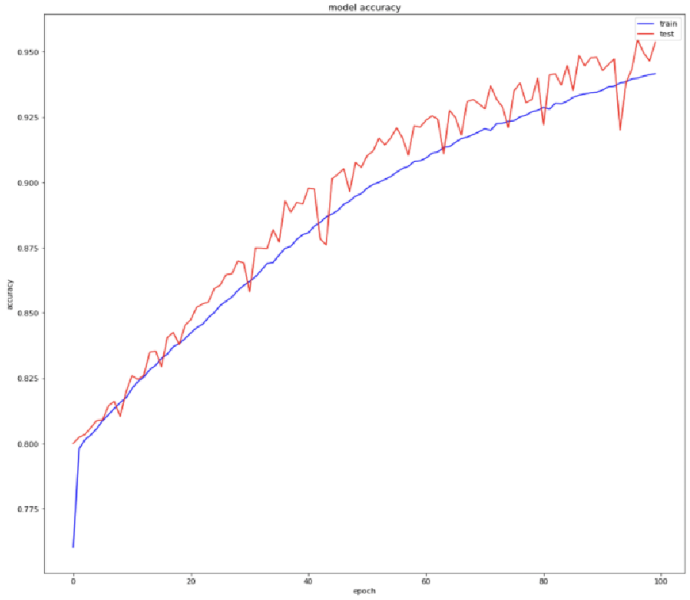


Рис 4.19. Графік метрики «Accuracy»

Для порівняння наведемо результати навченої під час описуваного експерименту системи. Вона має такі значення метрик:

- *Precision* = 0.952
- *Recall* = 0.951
- *MAP* = 0.2
- *nDCG* = 0.99

В цілому, можна вважати отримані результати не викликають сумнівів як розроблена рекомендаційна система. Значення метрики MAP у порівнянні вказують на те, що використання обробки природної мови для інформації про фільми справило позитивний ефект на результати передбачень системи.

Точність отриманих рекомендацій буде перевірена на тестовому наборі даних за описаними вище метриками RMSE, MSE і MAE. Щоб порівняти результати роботи моделі аналогічні дослідження на тому ж наборі даних були проведені на базових алгоритмах персоналізації окремо.

Знайдено значення метрик для алгоритмів розробленої гібридної моделі машинного навчання і порівняємо з нашим підходом (рис. 4.20).

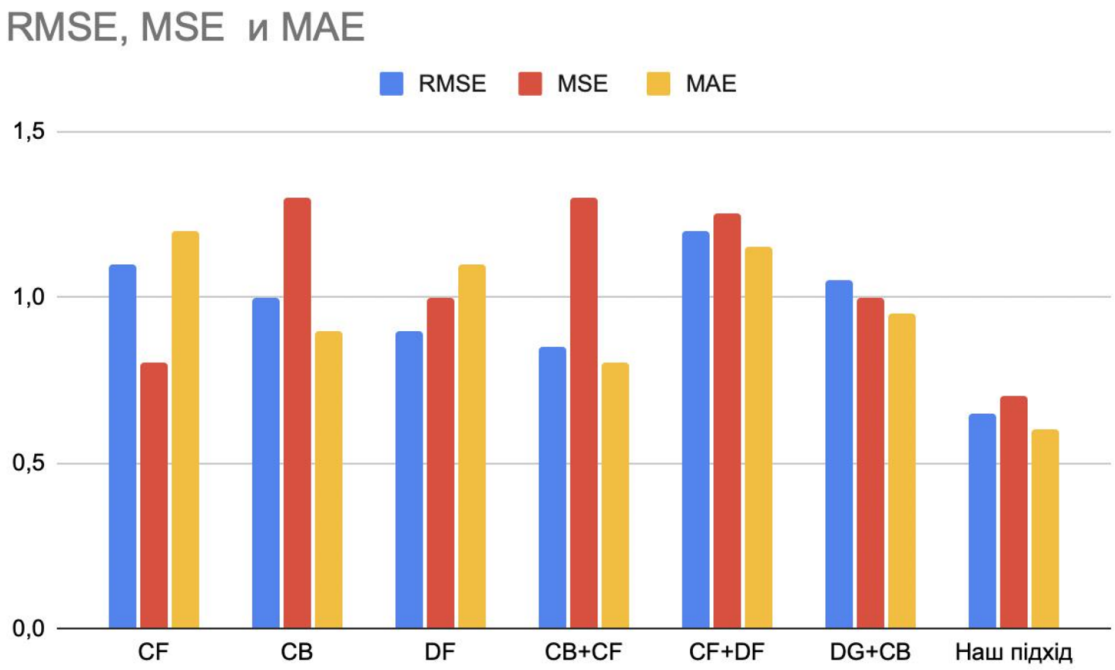


Рис. 4.20. Поріняння метрик RMSE, MSE і MAE для різних підходів

Відомо, що чим менше значення перелічених метрик тим більш точні рекомендації пропонує модель. Бачимо, що для обраного алгоритму ранжування, значення метрик RMSE, MSE, MAE зменшилось порівняно з content based filtering на 48.3%, 62.9%, 23.2%. відповідно, а з item-based collaborative filtering на 33.2%, 20.5%, 15.3% відповідно.

Проаналізувавши отримані результати бачимо, що модель машинного навчання використана для побудови рекомендаційної системи додатку більш точна, а отже і більш ефективна ніж протестовані базові моделі.

4.5. Висновки до розділу

Під час роботи над розділом були розроблені, проаналізовані та обрані відповідні програмні засоби для реалізації функцій додатку, а саме мова програмування, фреймворки, бібліотеки для створення інтерфейсу користувача, серверна частина додатку та реалізація алгоритмів персоналізації. Програмне забезпечення було створено за допомогою вибраних засобів розробки. В результаті роботи отримано сучасний, функціональний веб-додаток, який відповідає заданим програмним вимогам. Окремо були проведені експериментальні дослідження з використанням розробленої моделі машинного навчання.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Аналіз проблеми

Персоналізовані рекомендації ніколи не були такими важливими для підприємств, які прагнуть залучити та утримати клієнтів. За даними Accenture, дев'ять із 10 споживачів з більшою ймовірністю будуть робити покупки з брендами, які надають відповідні пропозиції та рекомендації.

Складнощі, з якими доводиться стикатися при побудові рекомендаційних систем, лише частково схожі на проблеми суміжних ML-областей. Такі системи багатоскладові, до того ж потребують постійного оновлення та покращення. З цього виникають три категорії проблем:

- модельні − складність побудови моделей у необхідності одночасно задовольняти різні критерії якості, а також у складності підготовки/інтерпретації наявних даних. Можливі проблеми:
  - Різноманітність типів фідбека (тривалість, лайки, кліки, додавання до обраного) та його неоднозначність. Клік, але не лайк і не покупка це що? Подобається результат користувачеві чи ні?
  - Різноманітність типів контенту (відео і картинки, їм, наприклад, тривалості взаємодії принципово не порівнянна).
  - Холодний старт користувача.
  - Холодний старт контенту.
  - Короткостроковість трендів та старіння контенту.
  - Глобальна популярність.
  - Оновлення рекомендацій (для користувачів з 100500 процесів, 100500 +1 мало змінює видачу).
  - Облік послідовності взаємодії (смаки змінюються).
  - Облік сезонності (на Новий рік та на 14 лютого у тренді будуть різні фільтри).

- Включення різних типів інформації (щоб враховувати як взаємодії, а й сам контент: картинки і текст, і навіть метадані користувача: геолокацію, платоспроможність та інше.

- Інфраструктурні проблеми – здебільшого виростають із необхідності постійного апдейту моделей. На практиці поширеними проблемами стають:
  - Масштабованість, швидкість видачі рекомендацій, кешування, навантаження на сервери.
  - Оновлення рекомендацій (швидке реагування на останні дії, новий контент).
  - Регулярний апдейт моделей.
  - Правильне проведення аб–тестів.
  - проблеми вибору метрик.

Але головна, мабуть, проблема, у тому, що, попри всю схожість завдань у цій галузі, в кожного сервісу залишається своя специфіка. У музичних сервісів композиції можуть слухати багаторазово, а фільм навіть найцікавіший рідко дивитися двічі. У Tiktok мільйони роликів і вони старіють з жахливою швидкістю. А в Airbnb необхідно оптимізувати не тільки задоволення туристів від хостів, а й хостів від туристів (тобто оптимізувати в обидві сторони). Важливо точно визначати критичні особливості сервісу рекомендаційної системи.

- Проблеми метрик – однією з таких критичних визначення речей є метрики якості системи. В ідеалі ми хотіли б прямо виміряти, наскільки користувача влаштовує сервіс, але це неможливо. Доводиться задовольнятися опосередкованими метриками, деякі з яких можуть конфліктувати. Наприклад, досить легко підвищити кількість переглядів (фільмів, роликів) з допомогою зменшення середнього часу перегляду. Але чи це буде добре? У багатьох навіть

простих метрик є свої підводні камені. Так, орієнтування на кількість кліків підвищує у видачі сміттєвий, клікбейтовий контент і не працює для стрічок, які просто читають, гортаючи без кліків.

5.2. Зацікавлені сторони

У результаті аналізу проблеми були визначені певні групи, яких може зацікавити запропонований проект:

- Бізнеси, що пропонують підписку на якийсь термін, так як Netflix, Spotify, YouTube, XBOX, Sony тощо.
- Бізнеси з сектору електронної комерції. Компанії та інтернет-магазини по типу Розетки, OLX, Amazon, тощо.
- Фінансові та страхові компанії.
- Програми-платформи онлайн навчання , таких як Coursera.
- Платформи соціальних мереж (рекомендації щодо соціальних зв'язків Facebook або Instagram).
- Дослідницькі статті, новини, акції та системи підтримки торгівлі.

Усі групи були проаналізовані за такими показниками:

- Зацікавленість у вирішенні проблеми;
- Вплив на рішення проблеми;
- методи розповсюдження запропонованого методу і ПЗ, що його реалізує.

Порівняння груп за цими параметрами в табл. 5.1

Таблиця 5.1

Аналіз зацікавлених сторін

Зацікавлена сторона	Рівень зацікавленості	Рівень впливу	Методи роботи
Бізнеси з підписками	9/10	5/10	Використання запропонованого алгоритму для більш якісної рекомендації контенту
Електронна комерція	9/10	9/10	Використання для підбору схожих товарів, оптимізації закупівель
Фінансові та страхові компанії	5/10	4/10	Використання для створення оптимального страхового полісу та кредитів
Платформи онлайн-навчання	9/10	9/10	Використання для підбору курсів за інтересами користувача, популяризації контенту
Платформи соціальних мереж	7/10	6/10	Використання для пропонуванню користувачу друзів, яких він можливо знає чи групи, які йому сподобаються
Дослідницькі статті, новини	5/10	7/10	Використання для проведення акцій, розсилок релевантним клієнтам

5.3. Аналіз рішення проблеми

Вирішення проблеми полягає у створенні власного методу вибору рекомендацій контенту для користувача. Створена програма повинна рекомендувати користувачеві актуальний новий і цікавий контент з урахуванням його переваг.

Створене програмне забезпечення повинно мати інтерфейс для роботи з ним, спосіб отримання рекомендацій контенту для користувачів, реалізацію запропонованого методу, відображення його результатів, перевірку рейтингів контенту, можливість змінювати ці рейтинги, а також зберігати всі необхідні елементи, інформацію в базі даних. В результаті ми отримуємо сервіс, схожий на існуючі, який відрізняється тим, що використовує інший спосіб підбору рекомендацій контенту для користувачів.

Розроблене програмне забезпечення змінює спосіб аналізу якості програмного забезпечення за допомогою комбінації вже існуючих підходів, а також нейронної мережі, метод також підвищує точність завдяки використанню нейронних мереж.

5.4. Бізнес-продукт. Основні характеристики бізнес-продукту

Результатом магістерської роботи є запропонований метод вирішення поставленої задачі та програмне забезпечення, що його реалізує.

Перший розділ цієї дипломної роботи містить детальний аналіз існуючих реалізацій вибору контенту, а другий розділ містить аналіз існуючих методологій для створення системи рекомендацій. Усі існуючі сервіси та методи використовують або дуже складну архітектуру, або комбінацію методів, потребують багато грошей і мають певні недоліки.

Перераховані вище причини призводять до необхідності розробки власного бізнес-рішення. Одним із таких рішень є запропонований метод і програмне забезпечення, яке його реалізує, яке вирішує описану проблему.

В результаті ми отримуємо програмне забезпечення, яке використовує метод, який використовує комбінацію існуючих методів, а також створює власну нейронну мережу для підвищення точності аналізу якості програмного забезпечення. Програмне забезпечення має відповідний інтерфейс для роботи з ним та відповідні функції та методи, які реалізують запропонований метод.

Загальний результат виводиться з результатів цих функцій і методів, який відображає якість проаналізованого програмного забезпечення.

Основні переваги запропонованого методу та програмного забезпечення:

- Вимагає менше технічних і фінансових ресурсів, ніж існуючі впровадження.
- Виправляє недоліки існуючих рішень.
- Спрощує процес вибору рекомендацій для користувача.
- Повністю модифікований.

Підсумовуючи, можна сказати, що користувач запропонованого методу та програмного забезпечення матиме готове та перевірене програмне забезпечення для аналізу якості програмного забезпечення, яке вирішить проблему неточного та неякісного відбору рекомендацій для користувачів з існуючих рішень та сервісів за допомогою нейронної системи. мережі. Крім того, стає можливим покращити ставлення користувачів до компаній або розробників аналізованого програмного забезпечення та збільшити їх фінансові показники та частку ринку.

5.5. Конкурентні переваги продукту

На ринку існують сервіси та методи, які підбирають рекомендації для користувача на основі його переваг. Але не існує жодного методу чи служби, що використовує таку комбінацію методів із використанням нейронної мережі. Можна сказати, що запропонований метод і програмне забезпечення, що його реалізує, є унікальними на ринку.

Отримане програмне забезпечення пропонує абсолютно новий спосіб оцінки якості програмного забезпечення, і оскільки цей метод використовує нейронні мережі, і якість набагато вища, ніж у інших реалізаціях.

5.6. Клієнти. Сегменти ринку споживання

Для того, щоб проект був реалізований з максимально можливою ефективністю, необхідно розділити всіх можливих клієнтів на певні групи. Це дозволяє точно визначити, хто і в якій мірі зацікавлений у пропонованому програмному забезпеченні.

Після ретельного аналізу ринку було визначено двох можливих клієнтів.

Одними з таких клієнтів є компанії, для яких рекомендації є основним продуктом, який дозволяє їм орієнтуватися на користувачів. Можливі клієнти це компанії, що надають підписку на відео, музику, ігри та онлайн-курси. Такі компанії повинні мати програмне забезпечення, яке допомагає її кінцевим користувачам використовувати пз з цікавим, новим контентом, постійно повертатися та в деяких випадках спрощувати проблему пошуку. Для такої компанії важлива не тільки якість програмного забезпечення, а й ставлення клієнтів до її компанії. Тому їй дуже важлива думка користувачів про її програмне забезпечення, що може безпосередньо вплинути на ставлення користувачів до її компанії загалом. Такі компанії постійно шукають та модернізують свої методи та способи аналізу рекомендацій користувачів. Отже, можна стверджувати, що така компанія є потенційним замовником запропонованого рішення.

Іншим таким клієнтом є компанії, для яких рекомендація контенту є додатковою функцією, корисною для користувачів. Причому і в першому, і в другому випадку їм дуже важлива не тільки якість програмного забезпечення, а й думка користувачів. Через це вони можуть бути потенційними клієнтами, оскільки їм важлива якість контенту, який вони надають. Однак їм можна порекомендувати ліцензію на використання розробленого програмного забезпечення, яке адаптоване для них, але не масштабоване для всіх продуктів.

5.7. Унікальна цінність пропозиції

Створене рішення пропонує абсолютно новий спосіб підбору рекомендацій для користувача та готове програмне забезпечення, яке його реалізує. Запропонований метод унікальний тим, що він використовує спліт кількох методів і нейронних мереж для рекомендацій, а не один метод, як багато існуючих рішень.

У результаті ми маємо програмне забезпечення, яке використовує абсолютно новий метод, а також використовує нейронні мережі, що значно підвищує точність і якість аналізу якості програмного забезпечення.

5.8. Бізнес-модель, доходи та витрати

Враховуючи детальний аналіз можливих замовників, було обрано 2 варіанти реалізації проекту:

- право використовувати платформи з підписками на контент – спрямоване на одноразову оплату компанією та надання права користування сервісом протягом 1 року та відповідну підтримку з боку розробників та підписання відповідного договору у разі порушення, від чого компанія втрачає це право;
- Ліцензія користувача – щомісячна оплата за користування сервісом.

Доходи та витрати звітуються у гривні з умовою, що реалізація проекту відбудеться у 2022 році на території України. Нижче наведено детальний аналіз обох варіантів.

5.8.1. Право на користування компаніям з підпискам

Ця опція призначена для надання повного доступу до методу та програмного забезпечення, що її реалізує, з можливістю його модифікації, а також відповідно підтримується розробниками. Усі компанії, які отримують таке право, отримують однакове програмне забезпечення з самого початку і можуть адаптувати його до своїх потреб, якщо це необхідно. Ця опція

особливо корисна для підприємств, які працюють за передплатою, тому витрати та доходи розраховувалися з урахуванням цього фактора. Вартість послуги для таких компаній буде досить високою, а тому не тільки якість послуги, але й інші фактори, наприклад дизайн програмного забезпечення, будуть дуже важливі.

У результаті, було прийнято рішення, що для реалізації проєкту для великої компанії знадобиться така команда: 3 розробники, 1 дизайнер, 2 менеджери, 3 працівників служби підтримки й такі витрати:

- оренда офісу;
- зарплата співробітникам;
- оплата за пз для служби підтримки;
- технічне забезпечення;
- ПЗ розробникам;
- ПЗ дизайнеру;
- Податки.

Оскільки право на користування надається на 1 рік, то і витрати були підраховані з урахуванням цього чиннику. Загальний результат можна побачити в табл. 5.2.

Таблиця 5.2

Витрати на проєкт при продажі на 1 рік ПЗ

Вид витрат	Вартість (грн)
Оренда офісу	300 000
Зарплата співробітникам	6 120 000
ПЗ для call-centre	10 000
Технічне забезпечення	250 000

Продовження табл. 5.2

ПЗ розробникам	20 000
ПЗ дизайнерам	5 000
Податки	363 000
Усього	6 825 750

Було прийнято рішення продавати право на користування за 1 500 000 грн. За таких умов, якщо 5 і більше компанії підпишуть договір, проєкт почне отримувати прибуток.

5.8.2 Ліцензія на користування

Ліцензія на використання дає доступ до процесу та програмного забезпечення без можливості модифікації та підтримки з боку розробників. Незважаючи на відсутність підтримки, помилки та недоліки програмного забезпечення виправлятись будуть. Кожен, хто купує підписку, отримує доступ до того самого програмного забезпечення без можливості налаштувати його відповідно до своїх потреб. Цей варіант хороший для нових компаній, які продають свої продукти і потребують рекомендаційної системи для збільшення доходу. З урахуванням цього фактору розраховувалися доходи та витрати.

У результаті, було прийнято рішення, що для реалізації проєкту для таких клієнтів знадобиться 20 днів, 1 розробник і 1 дизайнер і такі витрати:

- Зарплата співробітникам.
- Технічне забезпечення.
- ПЗ розробнику.
- Податки.

Загальний результат можна побачити в табл. 5.3.

Таблиця 5.3

Витрати на проєкт при продажі ліцензії на користування

Вид витрат	Вартість (грн)
Зарплата співробітникам	120 000
Технічне забезпечення	20 000
ПЗ розробникам	5 000
Податки	25 250
Усього	180 000

До того ж, треба зауважити, що після реалізації проєкту, необхідно буде щомісяця платити зарплату розробнику й податки, що в сумі 94 500 грн.

Було прийнято рішення продавати ліцензію за 5000 грн у місяць. За таких умов, якщо на старті буде придбано більш ніж 180 ліцензій, проєкт отримає назад витрачені кошти і якщо щомісячно буде купуватися більш ніж 83 ліцензій, проєкт буде отримувати прибуток.

5.9. Висновки до розділу

Запропоновано новий метод і програмне забезпечення, що його реалізує, які усувають недоліки існуючих систем і вирішують поставлену проблему. Проблема полягає в тому, що всі методи використовують спліт кількох методів, і немає методів і сервісів, які б аналізували якість програмного забезпечення за допомогою нейронних мереж.

Усі клієнти були проаналізовані. В результаті було отримано дві бізнес-моделі. Один пропонує продаж пз у рік користування програмою, а інший – продаж місячної ліцензії. Проаналізовано доходи та витрати обох варіантів.

Процедура та програмне забезпечення, яке її реалізує, спрямовані на аналіз якості програмного забезпечення шляхом вибору рекомендацій для

користувачів. Запропоноване рішення повністю вирішує цю проблему та створює абсолютно новий спосіб вибору рекомендацій для користувача з відповідними метриками та параметрами та використовує нейронні мережі для підвищення точності якості та аналізу. Програмний продукт підходить як для компаній, що підписуються, так і для малого бізнесу.

ВИСНОВКИ

Системи рекомендацій відкривають нові можливості отримання персоналізованої інформації в Інтернеті. Це також допомагає розв'язати проблему перевантаження інформацією, яка є дуже поширеним явищем у системах пошуку інформації, і дозволяє користувачам мати доступ до продуктів і послуг, які недоступні для користувачів у системі.

У цій роботі було проаналізовано наявні сервіси, які використовують рекомендаційні системи. Також була описана актуальність, сформована задача, яка буде проаналізована під час роботи над дисертацією. Було вказано призначення розробки рекомендаційної системи, цілі та задачі, які повинні бути досягнуті під час роботи з системою рекомендацій контенту.

Було обґрунтовано поняття рекомендаційних систем, які необхідні дані для її побудови, а також основні проблеми, які можуть виникнути під час її створення. Було визначено основні переваги та недоліки методів, зокрема колаборативної фільтрації, демографічну фільтрацію і фільтрації на основі вмісту та використання гібридної системи рекомендацій на добре відомомо наборі даних MovieLens. І визначено, що найкращим варіантом є комбінація існуючих методів для уникнення проблем з «холодним запуском», розрідженістю даних, недостатньою новизною, тощо. Звичайно, що для кожного продукту має будуватись окрема рекомендаційна система і комбінація різних методів в залежності від заданих цілей. Саме тому було описано обраний метод та необхідні дані, які будуть збиратись з веб застосунку. Також було описано основні типи метрик, які необхідні для оцінки ефективності, а також їх основні переваги та недоліки.

Було сформовано постановку задачі, обрано програмних забезпечень для реалізації веб застосунку, було детально описано функціональні можливості та вимоги до веб застосунку та технічного забезпечення. Було

опсано вискорівненву архітектуру програмного забезпечення і нейронної мережі.

Були проаналізовані та обрані відповідні програмні засоби для реалізації функцій додатку, а саме мова програмування, фреймворки, бібліотеки для створення інтерфейсу користувача, серверна частина додатку та реалізація алгоритмів персоналізації. Програмне забезпечення було створено за допомогою вибраних засобів розробки. В результаті роботи отримано сучасний, функціональний веб-додаток, який відповідає заданим програмним вимогам. Окремо були проведені експериментальні дослідження з використанням розробленої моделі машинного навчання.

Ринок ПЗ рекомендаійних систем було проанолізовано. Була описана поставлена проблема, спосіб її вирішення, конкурентні переваги запропонованого рішення і його клієнти. До того ж, у результаті детального аналізу можливих клієнтів були описані дві бізнес-моделі, їх доходи і витрати.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. G. Linden, B. Smith, and J. York. «Amazon.com Recommendations: Item–to–Item Collaborative Filtering». IEEE Internet Computing. [Електронний ресурс]. Режим доступу: https://www.cs.umd.edu/~samir/498/Amazon–Recommendations.pdf (2003).
2. B. Smith, G. Linden. «Two Decades of Recommender Systems at Amazon.com». IEEE Internet Computing. [Електронний ресурс]. Режим доступу: https://ieeecs–media.computer.org/assets/pdf/mic2017030012.pdf (2017).
3. A. Sharma, J.M. Hofman, D.J. Watts. «Estimating the Causal Impact of Recommendation Systems from Observational Data» Proc. 16th ACM Conf. Economics and Computation, 2015, pp. 453–470 (2015).
4. «Netflix prize» [Електронний ресурс]. Режим доступу: https://www.netflixprize.com/ (2009).
5. E. Damiani, M. Shang. «Recommender Systems». [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/topics/computer–science/recommender–systems (2020).
6. X. Amatriain, J. Basilico. «Netflix Recommendations: Beyond the 5 stars». Netflix Technology Blog. [Електронний ресурс]. Режим доступу: https://netflixtechblog.com/netflix–recommendations–beyond–the–5–stars–part–1–55838468f429 (2012).
7. C. Gomez–Uribe, N. Hunt. «The Netflix Recommender System: Algorithms, Business Value, and Innovation». [Електронний ресурс]. Режим доступу: https://dl.acm.org/doi/10.1145/2843948.
8. X. Amatriain, J. Basilico. «Netflix Recommendations: Beyond the 5 stars (Part 2)». Netflix Technology Blog. [Електронний ресурс]. Режим доступу: https://netflixtechblog.com/netflix–recommendations–beyond–the–5–stars–part–2–d9b96aa399f5.

9. R. Banik, T. Bozsolik. «The Movies Dataset». [Електронний ресурс]. Режим доступу: https://www.kaggle.com/rounakbanik/the-movies-dataset.
10. D. Sohier. «TMDB 5000 Movie Dataset». [Електронний ресурс]. Режим доступу: https://www.kaggle.com/tmdb/tmdb-movie-metadata.
11. E. Lehmann, G Casella. «Theory of Point Estimation (2nd ed.)». Springer. ISBN 0-387-98502-6. (1998).
12. «IMDB Help Center» [Електронний ресурс]. Режим доступу: https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#
13. N. Hug. «Similarities module». [Електронний ресурс]. Режим доступу: https://surprise.readthedocs.io/en/stable/similarities.html
14. S. Glen. «Correlation Coefficient: Definition, Formula». [Електронний ресурс]. Режим доступу: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula.
15. N. Hug. «Prediction algorithms module». [Електронний ресурс]. Режим доступу: https://surprise.readthedocs.io/en/stable/similarities.html.
16. «React documentation». [Електронний ресурс]. Режим доступу: https://reactjs.org/docs/getting-started.html.
17. «Redux documentation». [Електронний ресурс]. Режим доступу: https://redux.js.org/.
18. «React-Redux documentation» [Електронний ресурс]. Режим доступу: https://react-redux.js.org/.
19. «Semantic-UI documentation» [Електронний ресурс]. Режим доступу: https://react.semantic-ui.com/.
20. «Pandas documentation» [Електронний ресурс]. Режим доступу: https://pandas.pydata.org/docs/.
21. «Scikit-learn documentation» [Електронний ресурс]. Режим доступу: https://scikit-learn.org/stable/.

22. «Flask documentation» [Електронний ресурс]. https://flask.palletsprojects.com/en/2.0.x/ .
23. «MySQL documentation» [Електронний ресурс]. https://dev.mysql.com/doc/ (2021).
24. «PyJWT documentation» [Електронний ресурс]. https://pyjwt.readthedocs.io/en/latest/.
25. M. Nazim Uddin, J. Shrestha, and G.–S. Jo, "Enhanced content– based ltering using diverse collaborative prediction for movie recommendation," in Proceedings of the 2009 1st Asian Confer– ence on Intelligent Information and Database Systems, ACIIDS 2009, pp. 132–137, Viet Nam, April 2009.
26. A. Gunawardana and C. Meek, "A uni ed approach to building hybrid recommender systems," in Proceedings of the 3rd ACM Conference on Recommender Systems, RecSys'09, pp. 117–124, USA, October 2009.
27. K. Soni, R. Goyal, B. Vadera, and S. More, "A ree Way Hybrid Movie Recommendation Syste," International Journal of Computer Applications, vol. 160, no. 9, pp. 29–32, 2017.
28. G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a com– bined approach to recommend," in Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, pp. 105–112, USA, October 2014.
29. Y. Zhang, M. Chen, D. Huang, D. Wu, and Y. Li, "IDoctor: personalized and professionalized medical recommendations. Wireless Communications and Mobile Computing based on hybrid matrix factorization," Future Generation Computer Systems, vol. 66, pp. 30–35, 2017.
30. B. Pang, L. Lee, and S. Vaithyanathan, " umbs up?: sentiment classication using machine learning techniques," in Proceedings of the ACL–02 Conference on Empirical Methods in Natural Language Processing—Volume 10 (EMNLP '02), pp. 79– 86, Association for Computational Linguistics, Stroudsburg, Pa, USA, July 2002.

31. P. D. Turney, " umbs up or thumbs down?" in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 417–424, Philadelphia, Pennsylvania, July 2002.

32. J. R. Priester and R. E. Petty, " e Gradual reshold Model of Ambivalence: Relating the Positive and Negative Bases of Attitudes to Subjective Ambivalence," Journal of Personality and Social Psychology, vol. 71, no. 3, pp. 431–449, 1996.

33. H. Li, J. Cui, B. Shen, and J. Ma, "An intelligent movie recom− mendation system through group−level sentiment analysis in microblogs," Neurocomputing, vol. 210, pp. 164–173, 2016.

34. J. Sun, G. Wang, X. Cheng, and Y. Fu, "Mining a ective text to improve social media item recommendation," Information Processing & Management, vol. 51, no. 4, pp. 444–457, 2015.

35. Q. Diao, M. Qiu, C.−Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), pp. 193– 202, ACM, New York, NY, USA, August 2014.

36. Y.Zhang,"GroRec:AGroup−CentricIntelligentRecommender System Integrating Social, Mobile and Big Data Technologies," IEEE Transactions on Services Computing, vol. 9, no. 5, pp. 786– 795, 2016.

37. D. Liu, W. Xu, W. Du, and F. Wang, "How to choose appropriate experts for peer review: An intelligent recommendation method in a big data context," Data Science Journal, vol. 14, article no. 16, 2015.

38. W. Xu, J. Sun, J. Ma, and W. Du, "A personalized information recommendation system for RD project opportunity nding in big data contexts," Journal of Network and Computer Applica− tions, vol. 59, pp. 362– 369, 2016.

39. Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large–scale parallel collaborative ltering for the Net ix Prize," in Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management, pp. 337–348, 2008.

40. Z.–D.ZhaoandM.–S.Shang,"User–based collaborative filtering recommendation algorithms on hadoop," in Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, WKDD 2010, pp. 478–481, ailand, January 2010.

41. J. Sun, W. Xu, J. Ma, and J. Sun, "Leverage RAF to nd domain experts on research social network services: A big data analytics methodology with MapReduce framework," International Journal of Production Economics, vol. 165, pp. 185–193, 2015.

42. J. Jiang, J. Lu, G. Zhang, and G. Long, "Scaling–up item–based collaborative ltering recommendation algorithm based on Hadoop," in Proceedings of the 7th IEEE World Congress on Services, pp. 490–497, IEEE, Washington, DC, USA, July 2011.

43. S. Panigrahi, R. K. Lenka, and A. Stitipragyan, "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark," in Proceedings of the 7th International Conference on Ambient Systems, Networks and Technologies, ANT 2016 and the 6th International Conference on Sustainable Energy Information Technology, SEIT 2016, pp. 1000–1006, Spain, May 2016.

44. A. Wijayanto and E. Winarko, "Implementation of multi–criteria collaborative ltering on cluster using Apache Spark," in Proceedings of the 2nd International Conference on Science and Technology–Computer, ICST 2016, pp. 177–181, Indonesia, October 2016.

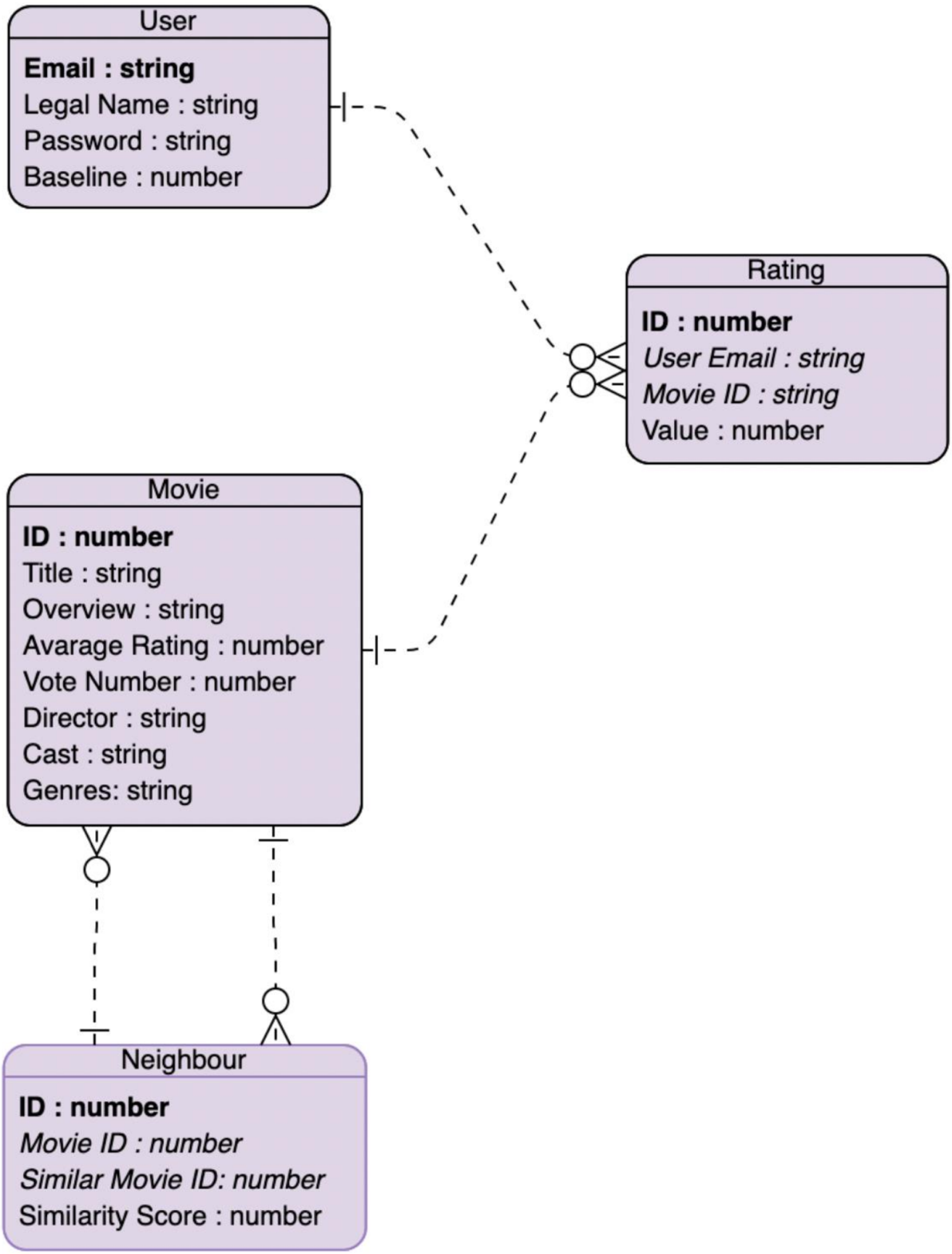
45. R. Feldman and I. Dagan, "Knowledge discovery in textual databases (KDT," in Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pp. 112–117, 1995.
46. A. H. Tan, "Text mining: the state of the art and the challenges," in Proceedings of the Paci c Asia Conference on Knowledge Discovery and Data Mining PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases, pp. 65–70, 1999.
47. A. Hotho, A. Nu ̈rnberger, and G. Paaß, "A brief survey of text mining," LDV Forum – GLDV Journal for Computational Linguistics and Language Technology, vol. 20, pp. 19–62, 2005.
48. H.–P.Zhang,H.–K.Yu,D.–Y.Xiong,andQ.Liu,"HHMM–based Chinese lexical analyzer ICTCLAS," in Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing (SIGHAN '03), pp. 184–187, Sapporo, Japan, July 2003.
49. Y.Wangand W.Xu,"LeveragingdeeplearningwithLDA–based text analytics to detect automobile insurance fraud," Decision Support Systems, vol. 105, pp. 87–95, 2018.
50. F. Wu, Y. Huang, Y. Song, and S. Liu, "Towards building a high– quality microblog–speci c Chinese sentiment lexicon," Decision Support Systems, vol. 87, pp. 39–49, 2016.
51. Y.Wang,W.Xu,andH.Jiang,"Usingtextminingandclustering to group research proposals for research project selection," in Proceedings of the 48th Annual Hawaii International Conference on System Sciences, HICSS 2015, pp. 1256–1263, USA, January 2015.

ДОДАТКИ

Додаток 1

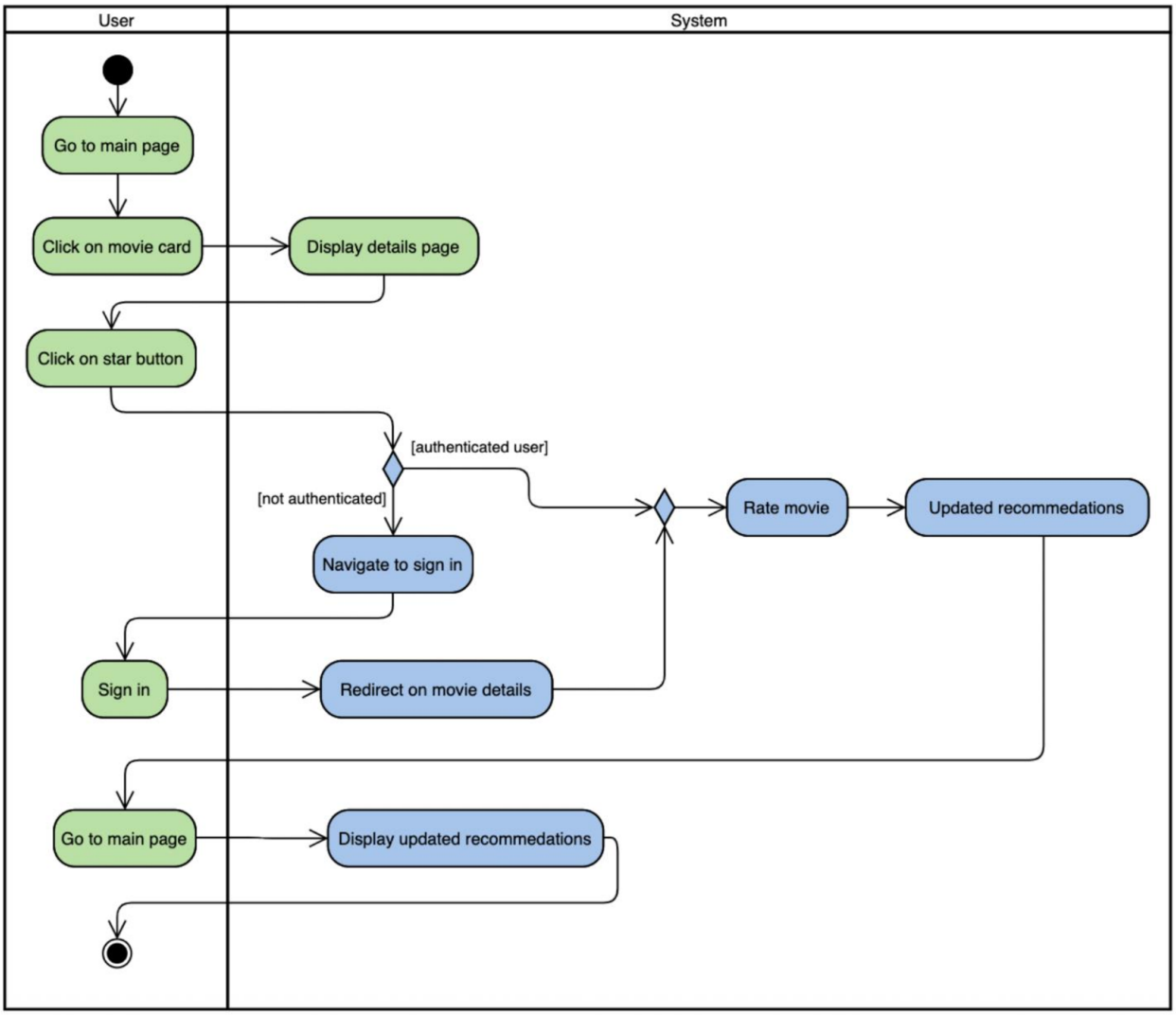
Копії графічних матеріалів

ER-модель бази даних



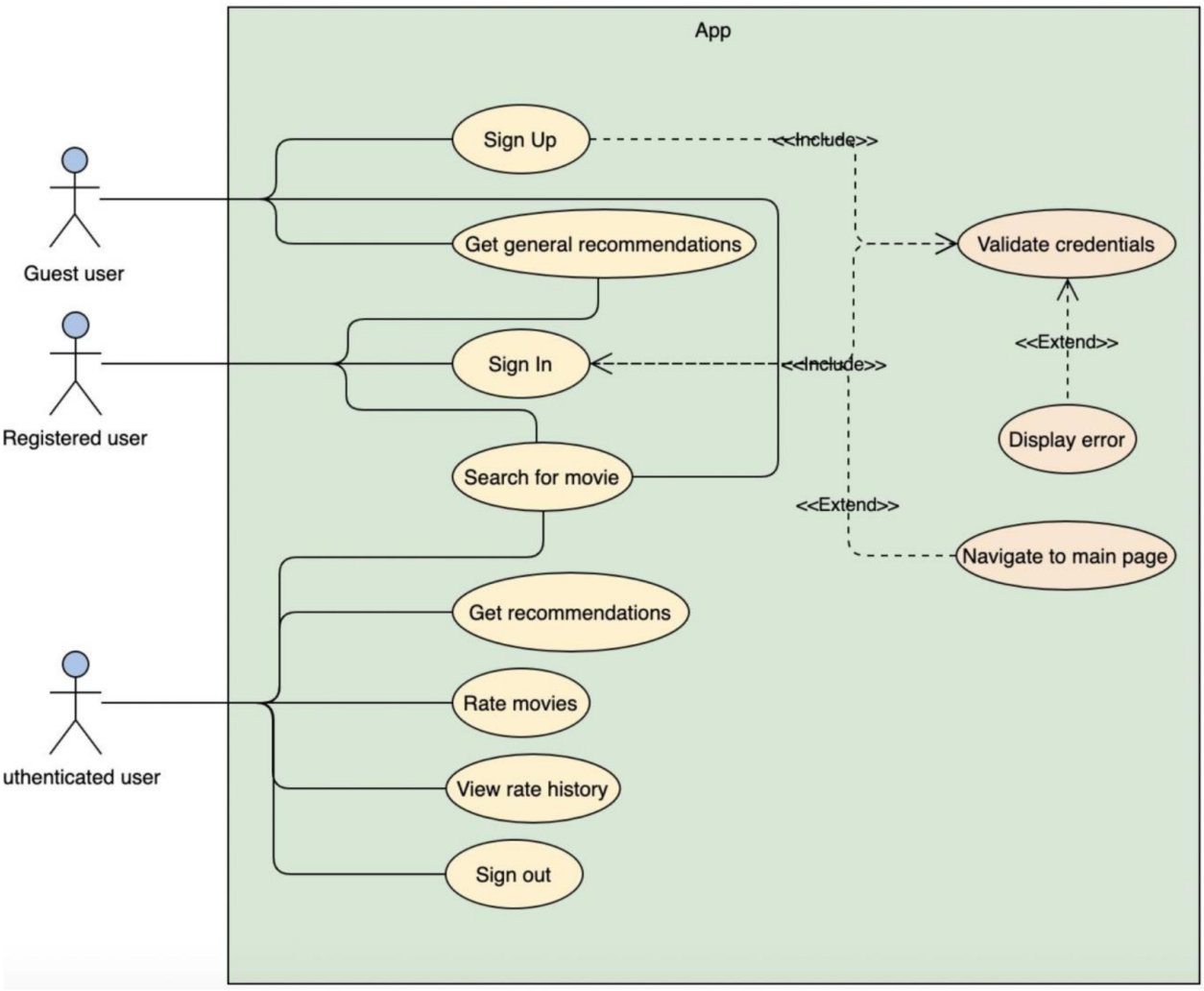
Фалілеєва Дар'я КП-11мп

Діаграма діяльності



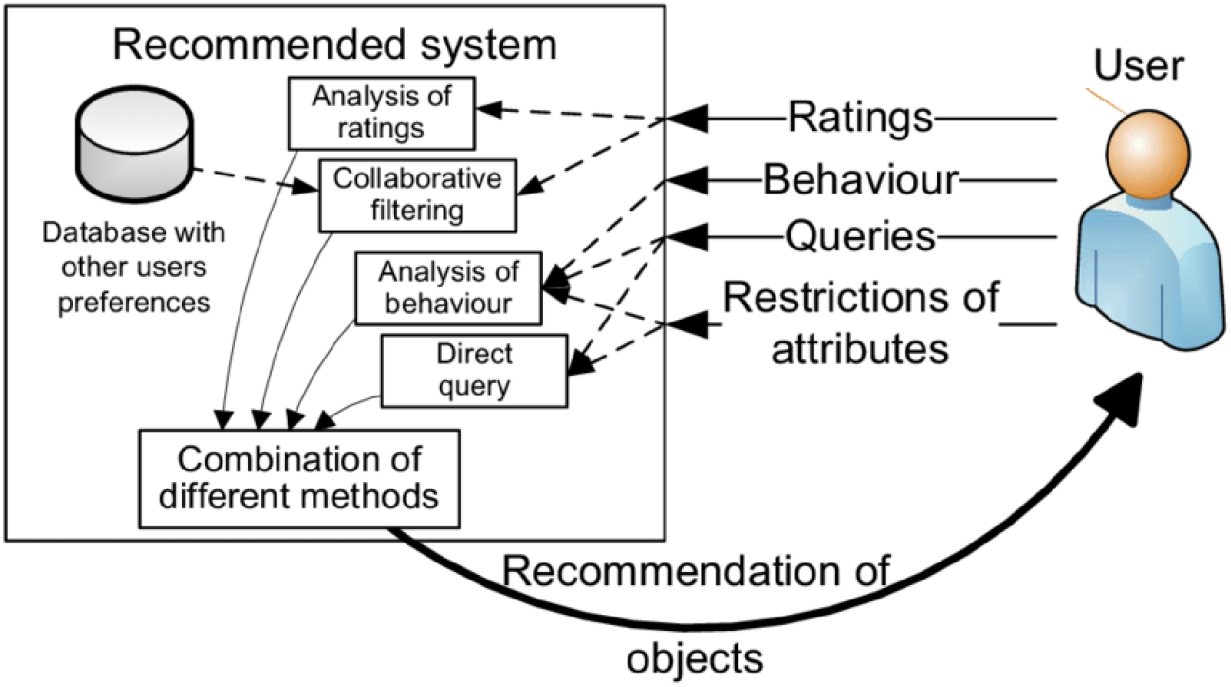
Фалілеєва Дар’я КП-11мп

Діаграма предедентів



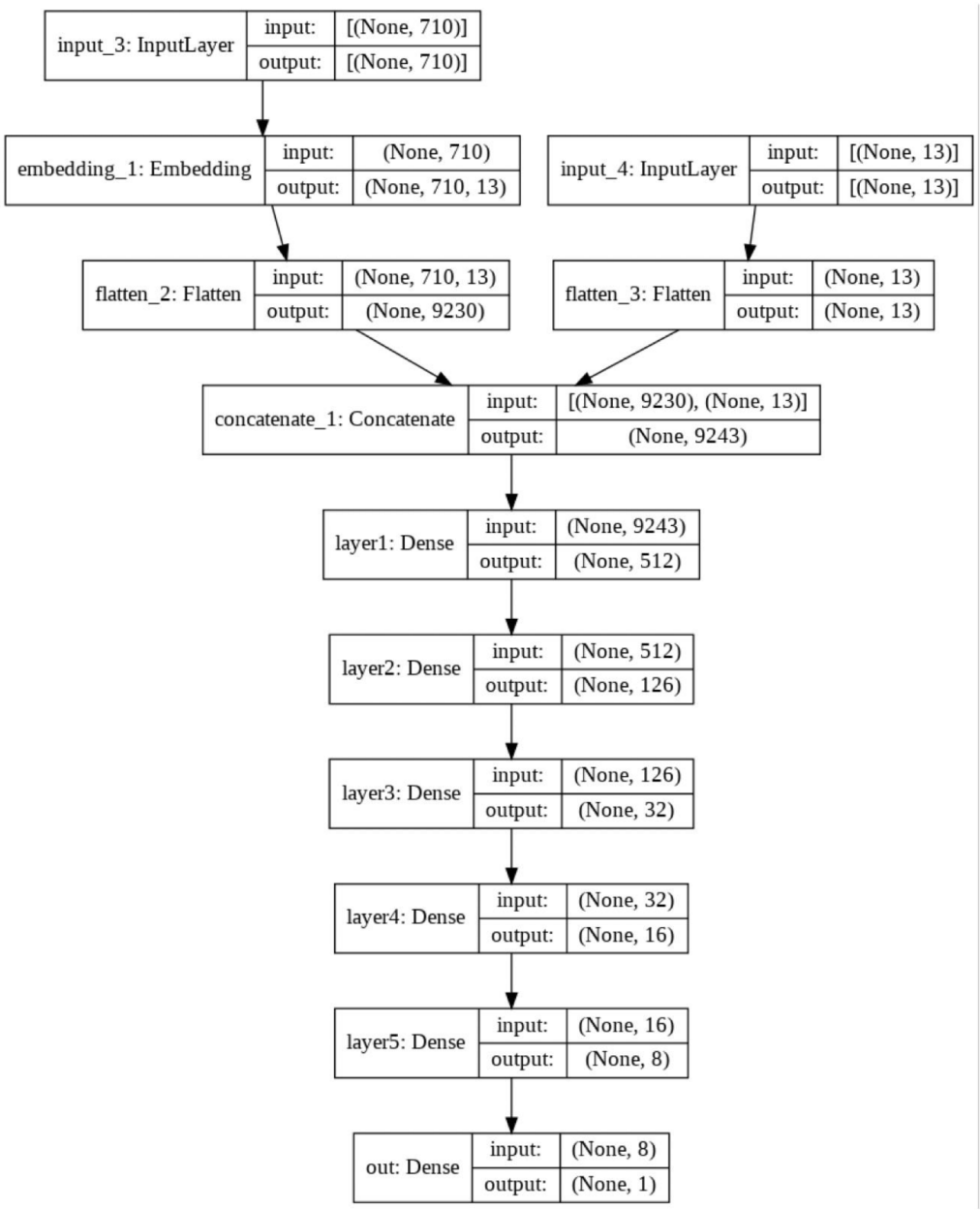
Фалілеєва Дар'я КП-11мп

Діаграма структури рекомендаційної системи



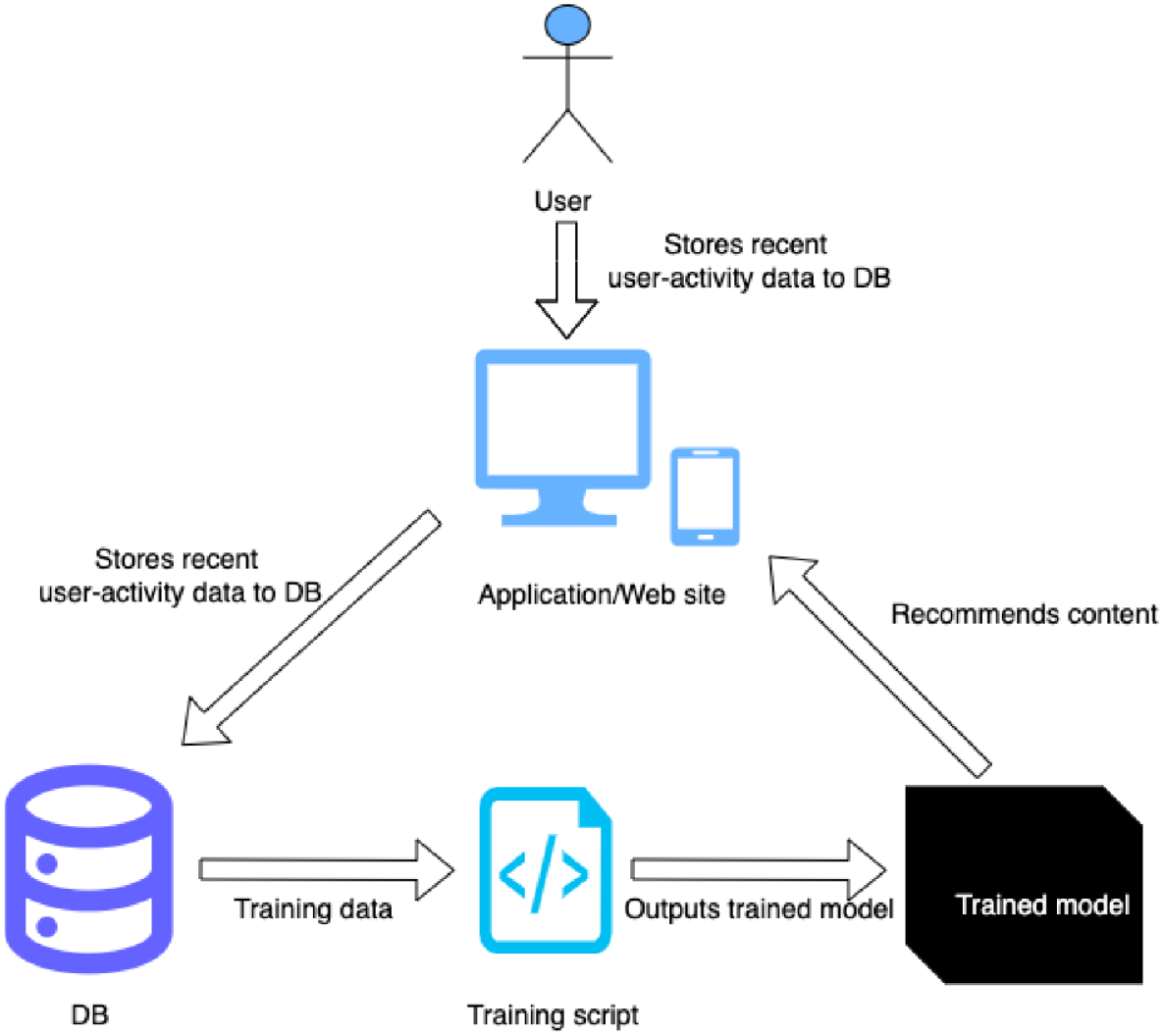
Фалілеєва Дар'я КП-11мп

Схема структури нейронної мережі



Фалілеєва Дар’я КП-11мп

Потік користувацьких даних на веб сайті



Фалілеєва Дар’я КП-11мп

Додаток 2

Лістинг програми

Лістинг 1.ML_LR.ipynb


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 97,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "<torch._C.Generator at 0x112002c30>"
      ]
     },
     "execution_count": 97,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "import os\n",
    "import numpy as np\n",
    "import random\n",
    "import torch\n",
    "import torch.nn as nn\n",
    "import torch.nn.functional as F\n",
    "from torch.autograd import Variable\n",
    "from torch import nn\n",
    "from torch.utils.data import Dataset, DataLoader\n",
    "torch.manual_seed(0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 98,
   "metadata": {},
   "outputs": [],
   "source": [
    "import argparse\n",
    "from time import time\n",
    "import pickle\n",
    "import scipy.sparse as sp\n",
    "# import pandas as pd\n",
    "np.random.seed(7)\n",
    "import math\n",
    "import heapq"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 99,
   "metadata": {},
   "outputs": [],
   "source": [
    "use_cuda = torch.cuda.is_available()\n",
    "device = torch.device(\"cuda:0\" if use_cuda else \"cpu\")\n",
```

```
   "# set the device as gpu or cpu depending upon the machine its running on"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 100,
  "metadata": {},
  "outputs": [],
  "source": [
   "epochs = 50\n",
   "batch_size = 256\n",
   "weight_decay = 0.00001\n",
   "num_negatives_pretrain = 4\n",
   "num_negatives_pasttest = 100\n",
   "learning_rate = 0.001\n",
   "dropout = 0.2\n",
   "optimizer = 'adam'\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 101,
  "metadata": {},
  "outputs": [],
  "source": [
   "# movie lens dataset has some positive interactions between user and movie means movie has been watched by user but doesnt\n",
   "# necessarily imply that user liked the movie. Below class, generates a sparse matrix with value 1 for positive interactions\n",
   "# and 0 otherwise. Initialzing the class with default parameters would return training data with 4 negative interaction for\n",
   "# for every positive interaction and test data with 100 negative interactions for every positive interaction\n",
   "class MovieLensDataset(Dataset):\n",
   "    \n",
   "    def __init__(self, file_name, num_negatives_train=5, num_negatives_test=100):\n",
   "        self.train_matrix = self.load_matrix(file_name + \".train.rating\")\n",
   "        self.n_users, self.n_items = self.train_matrix.shape\n",
   "                self.user_input, self.item_input, self.ratings = self.get_train_instances(self.train_matrix, num_negatives_train)\n",
   "        self.testRatings = self.load_list(file_name + \".test.rating\")\n",
   "        self.testNegatives = self.create_negatives(num_samples=num_negatives_test)\n",
   "        assert len(self.testRatings) == len(self.testNegatives)\n",
   "\n",
   "    def __len__(self):\n",
   "        return len(self.user_input)\n",
   "\n",
   "    def __getitem__(self, index):\n",
   "        return {'user_id': self.user_input[index],\n",
   "                'item_id': self.item_input[index],\n",
   "                'rating': self.ratings[index]}\n",
   "    def load_matrix(self, filename):\n",
   "        num_users, num_items = 0, 0\n",
   "        with open(filename, \"r\") as f:\n",
   "            line = f.readline()\n",
   "            while line != None and line != \"\":\n",
   "                arr = line.split(\"\\t\")\n",
```

```
"                u, i = int(arr[0]), int(arr[1])\n",
"                num_users = max(num_users, u)\n",
"                num_items = max(num_items, i)\n",
"                line = f.readline()\n",
"        mat = sp.dok_matrix((num_users+1, num_items+1), dtype=np.float32)\n",
"        with open(filename, \"r\") as f:\n",
"            line = f.readline()\n",
"            while line != None and line != \"\":\n",
"                arr = line.split(\"\\t\")\n",
"                user, item, rating = int(arr[0]), int(arr[1]), float(arr[2])\n",
"                if (rating > 0):\n",
"                    mat[user, item] = 1.0\n",
"                line = f.readline()\n",
"        return mat\n",
"\n",
"    def get_train_instances(self, train, num_negatives):\n",
"        user_input, item_input, ratings = [], [], []\n",
"        num_users, num_items = train.shape\n",
"        for (u, i) in train.keys():\n",
"            user_input.append(u)\n",
"            item_input.append(i)\n",
"            ratings.append(1)\n",
"            for t in range(num_negatives):\n",
"                j = np.random.randint(1, num_items)\n",
"                while (u, j) in train:\n",
"                    j = np.random.randint(1, num_items)\n",
"                user_input.append(u)\n",
"                item_input.append(j)\n",
"                ratings.append(0)\n",
"        return user_input, item_input, ratings\n",
"\n",
"    def load_list(self, filename):\n",
"        ratingList = []\n",
"        with open(filename, \"r\") as f:\n",
"            line = f.readline()\n",
"            while line != None and line != \"\":\n",
"                arr = line.split(\"\\t\")\n",
"                user, item = int(arr[0]), int(arr[1])\n",
"                ratingList.append([user, item])\n",
"                line = f.readline()\n",
"        return ratingList\n",
"\n",
"    def create_negatives(self, num_samples=100):\n",
"        negativeList = []\n",
"        for user_item_pair in self.testRatings:\n",
"            user = user_item_pair[0]\n",
"            item = user_item_pair[1]\n",
"            negatives = []\n",
"            for t in range(num_samples):\n",
"                j = np.random.randint(1, self.n_items)\n",
"                while (user, j) in self.train_matrix or j == item:\n",
"                    j = np.random.randint(1, self.n_items)\n",
"                negatives.append(j)\n",
"            negativeList.append(negatives)\n",
"        return negativeList\n",
"\n",
"    \n"
```

```
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 102,
  "metadata": {},
  "outputs": [],
  "source": [
   "# neural network with 3 hidden layers and input as concated embeddings for users and items. It return the probability of \n",
   "# that user will watch the given item\n",
   "class recommendationNetwork(nn.Module):\n",
   "    def __init__(self, n_users, n_items,embedding_dim):\n",
   "        super().__init__()\n",
   "        self.user_embed = torch.nn.Embedding(n_users, embedding_dim)\n",
   "        self.item_embed = torch.nn.Embedding(n_items, embedding_dim)\n",
   "        self.linear1 = nn.Linear(2*embedding_dim,32)\n",
   "        self.linear2 = nn.Linear(32,16)\n",
   "        self.linear3 = nn.Linear(16,8)\n",
   "        self.output = nn.Linear(8,1)\n",
   "#         self.D = torch.nn.Dropout(0.1)\n",
   "        \n",
   "    \n",
   "    def forward(self, inputUserItem):\n",
   "        users = inputUserItem['user_id']\n",
   "        items = inputUserItem['item_id']\n",
   "        user_embedding = self.user_embed(users)\n",
   "        item_embedding = self.item_embed(items)\n",
   "        # concatenate user and item embeddings to form input\n",
   "        input_embedding = torch.cat([user_embedding, item_embedding], 1)\n",
   "        hidden1 = (F.relu(self.linear1(input_embedding)))\n",
   "        hidden2 = (F.relu(self.linear2(hidden1)))\n",
   "        hidden3 = (F.relu(self.linear3(hidden2)))\n",
   "        output = torch.sigmoid(self.output(hidden3))\n",
   "        return output\n",
   "\n",
   "    def predict(self, inputUserItem):\n",
   "        # return the score, inputs and outputs are numpy arrays\n",
   "        for key in inputUserItem:\n",
   "            if type(inputUserItem[key])!= type(None):\n",
   "                                inputUserItem[key] = torch.from_numpy(inputUserItem[key]).to(dtype=torch.long, device=device)\n",
   "        output_scores = self.forward(inputUserItem)\n",
   "        return output_scores.cpu().detach().numpy()\n",
   "\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 103,
  "metadata": {},
  "outputs": [],
  "source": [
   "import math\n",
   "import heapq\n",
   "# function to evaluate the performance of trained model on complete test dataset using the metric of hit-ratio. \n",
```

```
   "\n",
   "def evaluate_model(model,testRatings,testNegatives,topK: int):\n",
   "    hitratio = []\n",
   "    for idx in range(len(testRatings)):\n",
   "    \n",
   "        itemsList = testNegatives[idx]\n",
   "        u = testRatings[idx][0]\n",
   "        positiveItem = testRatings[idx][1]\n",
   "        itemsList.append(positiveItem)\n",
   "        map_item_score = {}\n",
   "        users = np.full(len(itemsList), u, dtype='int32')\n",
   "\n",
   "        data = {\n",
   "            'user_id': users,\n",
   "            'item_id': np.array(itemsList),\n",
   "        }\n",
   "        predictions = model.predict(data)\n",
   "        for i in range(len(itemsList)):\n",
   "            item = itemsList[i]\n",
   "            map_item_score[item] = predictions[i]\n",
   "        ranklist = heapq.nlargest(topK, map_item_score, key=map_item_score.get)\n",
   "        hr = getHitRatio(ranklist, positiveItem)\n",
   "        hitratio.append(hr)\n",
   "    return hitratio\n",
   "\n",
   "def getHitRatio(ranklist, positiveItem):\n",
   "    for item in ranklist:\n",
   "        if item == positiveItem:\n",
   "            return 1\n",
   "    return 0"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 104,
  "metadata": {},
  "outputs": [],
  "source": [
   "# reading the dataset from local and creating movielensdataset class to generate a spare train matrix
and test datastet\n",
   "dataset                                                                                    =
MovieLensDataset(\"/Users/shiprajain/Desktop/RecommenderSystem/movielens/movielens\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 105,
  "metadata": {},
  "outputs": [],
  "source": [
   "# pytorch's dataloader shuffles the datapoints and divide it in batches of fixed size for batch training \n",
   "training_data_generator      =      DataLoader(dataset,      batch_size=batch_size,      shuffle=True,
num_workers=0)"
  ]
 },
 {
  "cell_type": "code",
```

```
  "execution_count": 106,
  "metadata": {},
  "outputs": [],
  "source": [
   "train = dataset.train_matrix\n",
   "num_users, num_items = train.shape\n",
   "model = recommendationNetwork(num_users, num_items,8)\n",
   "model.to(device)\n",
   "loss_fn = torch.nn.BCELoss()\n",
   "optimizer = torch.optim.Adam(model.parameters(), weight_decay=weight_decay)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 107,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream"
   }
  ],
  "source": [
   "# training the model with adam optimizer and binary cross entropy loss function and testing its performance for every epoch\n",
   "for epoch in range(epochs):\n",
   "    epoch_loss = []\n",
   "    model.train()\n",
   "    for data in training_data_generator:\n",
   "        for key in data:\n",
   "            if type(data[key]) != type(None):\n",
   "                data[key] = data[key].to(dtype = torch.long, device = device)\n",
   "        prediction = model(data)\n",
   "        rating = data['rating']\n",
   "        rating = rating.float().view(prediction.size())  \n",
   "        loss = loss_fn(prediction, rating)\n",
   "        optimizer.zero_grad()\n",
   "        loss.backward()\n",
   "        optimizer.step()\n",
   "        epoch_loss.append(loss.item())\n",
   "    hitRatios = evaluate_model(model,dataset.testRatings,dataset.testNegatives,10)\n",
   "    print(\"epoch : \", epoch)\n",
   "    print(\"average loss : \",np.mean(epoch_loss))\n",
   "    print(\"average hit ratio : \",np.mean(hitRatios))\n",
   "    \n",
   "\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
 }
],
"metadata": {
```

```
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.7.0"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```

Додаток 3

Копія презентації