Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка рекомендаційної системи на базі нейронних мереж
(тема)

Виконав:
студент 2 курсу, групи КІТм-20-1
Макогон Ю.О.
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні інтелектуальні технології

Керівник проф. Руденко О.Г.
(посада, прізвище, ініціали)

Допускається до захисту _____
(підпис)

Зав. кафедри КІТС _____ проф. Руденко О.Г.
(підпис)

2021 р.

Харківський національний університет радіоелектроніки

Факультет	Комп’ютерної інженерії та управління
Кафедра	Комп’ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп’ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп’ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Макогону Ю.О.
(прізвище, ініціали)

1. Тема роботи (проекту) Розробка рекомендаційної системи на базі нейронних мереж

затверджена наказом по університету від “ 08 ” листопада 2021 р. № 1666Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вхідні дані до роботи
Документація React.js
Документацій Python
Документація Django
Редактор visual studio code
Особливості реалізації рекомендаційних систем
Документація PostgreSQL

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз предметної області
Постановка задачі
Вибір засобів розробки веб-сайтута рекомендаційної системи
Розробка бази даних інформаційної системи
Реалізація програмного забезпечення
Тестування системи
Аналіз результатів роботи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри)

8 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	08.11.2021	
2	Аналіз проблемної галузі, постановка задачі,	09.11.2021 – 13.11.2021	
	вибір інструментальних засобів		
3	Формування переліку вимог до програми	14.11.2021 – 18.11.2021	
4	Збір даних для системи	19.11.2021 – 20.11.2021	
5	Розробка моделі штучного інтелекту	21.11.2021 – 27.11.2021	
6	Проведення тестування системи	28.11.2021 – 30.11.2021	
7	Оформлення пояснювальної записки	01.12.2021 – 09.12.2021	
8	Перевірка виконаного проекту керівником	10.12.2021	
9	Захист проекту	16.12.2021 – 17.12.2021	

Дата видачі завдання ____ _____ 2021 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Руденко О.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 16 рис., 2 табл., 2 дод., 14 джерел.

НЕЙРОННА МЕРЕЖА, РЕКОМЕНДАЦІЙНА СИСТЕМА, КЛІЄНТ, СЕРВЕР, БАЗА ДАНИХ, МАТРИЦЯ, ФУНКЦІЯ АКТИВАЦІЇ.

Метою кваліфікаційної роботи є розробка рекомендаційної системи.

У ході виконання кваліфікаційної роботи було розроблено веб-сайт, що використовує алгоритм рекомендацій, побудований на базі нейронних мереж.

ABSTRACT

Qualification project: 75 pages, 16 figures, 2 tables, 2 appendice, 14 sources.

NEURAL NETWORK, RECOMMENDATION SYSTEM, CLIENT, SERVER, DATABASE, MATRIX, ACTIVATION FUNCTION.

The major goal of the qualification work is developing of recommendation system.

In order to implementation of this work was developed a website, which uses recommendation algorithm based on neural networks technology.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Розробка рекомендаційної системи на базі нейронних мереж

(тема)

Виконав:
студент 2 курсу, групи КІТм-20-1
Макогон Ю.О.
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні інтелектуальні технології
Керівник проф. Руденко О.Г.
(посада, прізвище, ініціали)

2021 р.

ВСТУП

Майже кожен з нас може проводити години блукаючи інтернетом в пошуках чогось необхідного. Це можуть бути деталі для автомобілів, книга, смартфон, кіно для перегляду вільним вечором і багато чого іншого. В сучасному світі багатьом знайома ситуація, коли ти заходиш на Aliexpress в пошуках лампи для кімнати і ловиш себе на перегляді сторінок з милими шкарпетками. Дорогоцінний час може бути витрачено на більш корисні речі.

Раніше, наприклад, люди ходили до магазинів з відеокасетими в пошуках кіно. Але кількість касет була обмеженої розмірами магазину. Зараз же інтернет дає людям доступ до відносно необмеженого джерела кінофільмів. Але, одночасно з появою такої скарбниці інформації, на людину щоденно почало виливатися потік інформації, потрібної і непотрібної, з яким досить таки складно впоратись. Щоб якось зменшити цей потік, всю цю інформацію потрібно фільтрувати. Більше того, компанії, які займаються поширенням контенту самі зацікавлені в методах підбору інформації, адже сучасні користувачі схильні оцінювати сервіс за лічені хвилини. Якщо за цей час він не знайде нічого, що могло б його зацікавити, то просто закриє вкладку браузеру або додаток для смартфону і вирушить в пошуках іншого. Адже він розуміє, що в інтернеті існує безліч альтернатив. Сучасному користувачеві вже не цікаві сервіси з купою реклами, поганою розміткою і незручним дизайном.

ОСНОВНА ЧАСТИНА

Власники кожного сервісу зацікавлені в тому, щоб кожен користувач залишався на їх сайті або додатку як можна довше. Як же пробудити в ньому інтерес? Відомо, що юзер буде більше зацікавлений в користуванні сайтом, який ніби спілкується особисто з ним, а не просто поводить себе як робот, який доносить інформацію. І тут виникає питання: як же спроектувати свій сервіс таким чином? Як навчити його вирізняти індивідуальність кожного? Відповіддю на ці питання можуть стати рекомендаційні системи. Це системи, які здатні прогнозувати перевагу певного набору елементів для конкретного користувача та рекомендувати найбільш популярні елементи. За їх допомогою можна фільтрувати інформацію індивідуально для кожного. До того ж, цих фільтрів може буди досить таки багато: геолокація запитувача, його вік, стать, вподобання, порівняння з іншими юзерами і т. д.

Але як же навчити нашу програму ідентифікувати кожного користувача та ще й вибирати контент особисто для нього? Для цього нам потрібно звернутися до сучасних технологій і вибрати ту, яку можна описати словом “навчається”. Саме такою технологією є нейронні мережі. За допомогою них програми навчають розпізнавати об’єкти на зображеннях і відео, обробляти і навіть створювати їх, працювати з аудіо, виконувати роль оператора call-центру, перекладати текст і багато іншого. Ідея полягає в тому, щоб максимально близько змоделювати роботу людської нервової системи - а саме, її здатності до навчання і виправлення помилок. У цьому полягає головна особливість будь-якої нейронної мережі - вона здатна самостійно навчатися і діяти на підставі попереднього досвіду, з кожним разом роблячи все менше помилок. Нейронна мережа імітує не тільки діяльність, а й структуру нервової системи людини, яка складається з нейронів.

Нейронні мережі успішно застосовуються в самих різних областях - бізнесі, медицині, техніці, геології, фізиці. Нейронні мережі ввійшли в практику

скрізь, де потрібно вирішити завдання прогнозування, класифікації або управління. Такий вражаючий успіх визначається кількома причинами:

- Багаті можливості. Нейронні мережі - виключно потужний метод моделювання, що дозволяє відтворювати надзвичайно складні залежності. Протягом багатьох років лінійне моделювання було основним методом моделювання в більшості областей, оскільки для нього добре розроблені процедури оптимізації. Крім того, нейронні мережі справляються з "прокляттям розмірності", яке не дозволяє моделювати лінійні залежності в разі великого числа змінних;

- Простота у використанні. Нейронні мережі навчаються на прикладах. Користувач нейронної мережі підбирає представницькі дані, а потім запускає алгоритм навчання, який автоматично сприймає структуру даних. При цьому від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати і готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніше, ніж, наприклад, при використанні традиційних методів статистики.

Нейронні мережі привабливі з інтуїтивної точки зору, бо вони засновані на примітивній біологічній моделі нервових систем. В майбутньому розвиток таких нейро-біологічних моделей може привести до створення дійсно мислячих комп'ютерів. Тим часом вже "прості" нейронні мережі, які будує система ST Neural Networks, є потужною зброєю в арсеналі фахівця з прикладної статистики.

Математичні методи, які застосовуються при розробці рекомендаційних систем, можна розбити на дві групи: методи коллаборативной фільтрації (collaborative filtering) і тематичні методи (content-based, information filtering). Звичайно, можливе одночасне використання методів двох груп (hybrid prediction). Зазвичай користь від товару представлена певною кількістю балів, що демонструють, наскільки конкретному користувачеві сподобався конкретний товар. У найзагальнішому вигляді проблема зводиться до присвоєння тієї чи іншої оцінки товару, ще не відомого покупцеві. Очевидно, така оцінка дається

виходячи з аналізу попередніх переваг даного покупця або будь-якої інший інформації про нього. Після того як система передбачає оцінки для ще не відомих споживачеві товарів: ті з них, які отримують найвищі оцінки рекомендуються потенційному споживачеві.

Коллаборативна фільтрація аналізує природу кожного елемента. Наприклад, рекомендує певного письменника користувачеві, виконуючи обробку попередніх оцінок його творчості користувачами. З іншого боку, коллаборативна фільтрація не вимагає жодної інформації про об’єкти рекомендацій або користувачів. Він рекомендує елементи на основі їх попередньої поведінки.

В процесі даної роботи буде написано веб сайт за допомогою мови програмування Python і фреймворку для створення web-серверу Django. Вибір данної мови обгрунтовано тим, що переважна більшість готових рішень для використання нейронних мереж реалізована саме на цій мові. Для створення клієнтської частини буде використано HTML, CSS і мову програмування Javascript, а саме фреймворк React.js, призначений для розробки single page applications. Це означає, що клієнт не буде звертатися до серверу кожен раз, коли йому потрібно буде оновити сторінку або загрузити якусь нову. З серверу буде надіслано лише один HTML файл, який потім буде просити в нього лише данні, використовуючи REST API.

ВИСНОВКИ

Для виконання кваліфікаційної роботи були вивчені основні методи і алгоритми рекомендаційної системи. На їх основі реалізований прототип веб-сайту для побудови профілю та вибору кінофільму.

В ході даної роботи були вирішені наступні завдання:

- було досліджені способи і алгоритми рекомендаційних систем, їх недоліки та особливості;
- було досліджено особливості використання та імплементації нейронних мереж;
- здобуто навики використання React.js, Django, Python, PostgreSQL.
- спроектовано базу даних для збереження інформації про об’єкти, які використовуються для розрахунку рекомендацій;
- розроблено алгоритм рекомендацій;
- описані додаткові функції системи;
- розроблено прототип веб-сайту для пошуку та оцінювання фільмів;

В майбутньому планується розширити алгоритм рекомендацій за допомогою класифікації тексту коментарів нейронними мережами.

Ключові слова: штучна нейронна мережа, машинне навчання, глибоке навчання, згорткова нейронна мережа, рекурентна нейронна мережа, довга короткострокова пам’ять.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Макогон Ю.О. Рекомендаційна система на базі нейронних мереж // Матеріали 25-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». – 2021. – 232 с.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Networks

COCO – Common Objects in Context

DNN – Deep Neural Network

GPU – Graphics processing unit

LSTM – Long short-term memory

RNN – Recurrent Neural Network

БД — база даних

КФ — коллаборативна фільтрація

НН – нейронна мережа

РС – рекомендаційна система

ШІ – Штучний інтелект

ВСТУП

Майже кожен з нас може проводити години блукаючи інтернетом в пошуках чогось необхідного. Це можуть бути деталі для автомобілів, книга, смартфон, кіно для перегляду вільним вечором і багато чого іншого. В сучасному світі багатьом знайома ситуація, коли ти заходиш на Aliexpress в пошуках лампи для кімнати і ловиш себе на перегляді сторінок з милими шкарпетками. Дорогоцінний час може бути витрачено на більш корисні речі.

Раніше, наприклад, люди ходили до магазинів з відеокасетими в пошуках кіно. Але кількість касет була обмеженої розмірами магазину. Зараз же інтернет дає людям доступ до відносно необмеженого джерела кінофільмів. Але, одночасно з появою такої скарбниці інформації, на людину щоденно почало виливатися потік інформації, потрібної і непотрібної, з яким досить таки складно впоратись. Щоб якось зменшити цей потік, всю цю інформацію потрібно фільтрувати. Більше того, компанії, які займаються поширенням контенту самі зацікавлені в методах підбору інформації, адже сучасні користувачі схильні оцінювати сервіс за лічені хвилини. Якщо за цей час він не знайде нічого, що могло б його зацікавити, то просто закриє вкладку браузеру або додаток для смартфону і вирушить в пошуках іншого. Адже він розуміє, що в інтернеті існує безліч альтернатив. Сучасному користувачеві вже не цікаві сервіси з купою реклами, поганою розміткою і незручним дизайном.

Власники кожного сервісу зацікавлені в тому, щоб кожен користувач залишався на їх сайті або додатку як можна довше. Як же пробудити в ньому інтерес? Відомо, що юзер буде більше зацікавлений в користуванні сайтом, який ніби спілкується особисто з ним, а не просто поводить себе як робот, який доносить інформацію. І тут виникає питання: як же спроектувати свій сервіс таким чином? Як навчити його вирізняти індивідуальність кожного? Відповіддю на ці питання можуть стати рекомендаційні системи. Це системи які здатні прогнозувати перевагу певного набору елементів для конкретного користувача

та рекомендувати найбільш популярні елементи. За їх допомогою можна фільтрувати інформацію індивідуально для кожного. До того ж, цих фільтрів може буди досить таки багато: геолокація запитувача, його вік, стать, вподобання, порівняння з іншими юзерами і т. д.

Але як же навчити нашу програму ідентифікувати кожного користувача та ще й вибирати контент особисто для нього? Для цього нам потрібно звернутися до сучасних технологій і вибрати ту, яку можна описати словом “навчається”. Саме такою технологією є нейронні мережі. За допомогою них програми навчають розпізнавати об’єкти на зображеннях і відео, обробляти і навіть створювати їх, працювати з аудіо, виконувати роль оператора call-центру, перекладати текст і багато іншого. Ідея полягає в тому, щоб максимально близько змоделювати роботу людської нервової системи - а саме, її здатності до навчання і виправлення помилок. У цьому полягає головна особливість будь-якої нейронної мережі - вона здатна самостійно навчатися і діяти на підставі попереднього досвіду, з кожним разом роблячи все менше помилок. Нейронна мережа імітує не тільки діяльність, а й структуру нервової системи людини, яка складається з нейронів.

Нейронні мережі успішно застосовуються в самих різних областях - бізнесі, медицині, техніці, геології, фізиці. Нейронні мережі ввійшли в практику скрізь, де потрібно вирішити завдання прогнозування, класифікації або управління. Такий вражаючий успіх визначається кількома причинами:

- Багаті можливості. Нейронні мережі - виключно потужний метод моделювання, що дозволяє відтворювати надзвичайно складні залежності. Протягом багатьох років лінійне моделювання було основним методом моделювання в більшості областей, оскільки для нього добре розроблені процедури оптимізації. Крім того, нейронні мережі справляються з "прокляттям розмірності", яке не дозволяє моделювати лінійні залежності в разі великого числа змінних;

- Простота у використанні. Нейронні мережі навчаються на прикладах. Користувач нейронної мережі підбирає представницькі дані, а потім запускає

алгоритм навчання, який автоматично сприймає структуру даних. При цьому від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати і готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніше, ніж, наприклад, при використанні традиційних методів статистики.

Нейронні мережі привабливі з інтуїтивної точки зору, бо вони засновані на примітивній біологічній моделі нервових систем. В майбутньому розвиток таких нейро-біологічних моделей може привести до створення дійсно мислячих комп'ютерів. Тим часом вже "прості" нейронні мережі, які будує система ST Neural Networks, є потужною зброєю в арсеналі фахівця з прикладної статистики.

В процесі даної роботи буде написано веб сайт за допомогою мови програмування Python і фреймворку для створення web-серверу Django. Вибір данної мови обгрунтовано тим, що переважна більшість готових рішень для використання нейронних мереж реалізована саме на цій мові. Для створення клієнтської частини буде використано HTML, CSS і мову програмування Javascript, а саме фреймворк React.js, призначений для розробки single page applications. Це означає, що клієнт не буде звертатися до серверу кожен раз, коли йому потрібно буде оновити сторінку або загрузити якусь нову. З серверу буде надіслано лише один HTML файл, який потім буде просити в нього лише данні, використовуючи REST API.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Описання предметної області

Аудиторія інтернету на території СНГ складає приблизно 175 млн. чоловік і ця цифра стрімко збільшується, все більше людей створюють трафік і об’єми медіаресурсів ростуть. В тому числі й кіно. Щороку в США випускається приблизно 700 кінострічок.

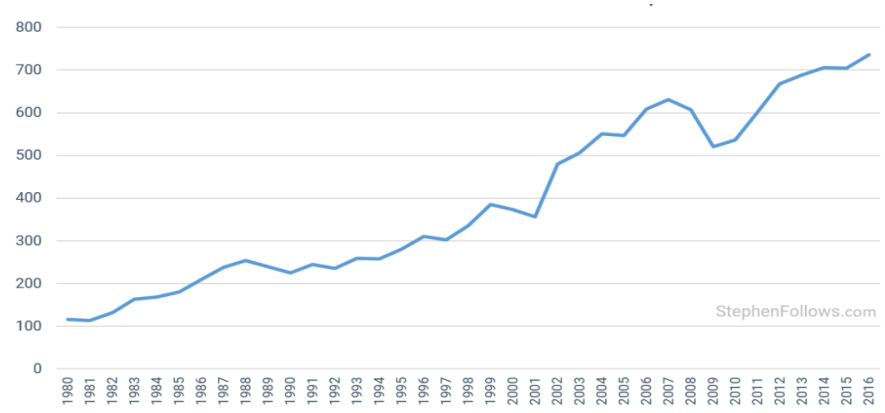

Рисунок 1.1 – Кількість фільмів, випущених в США, 1980-2017

Великобританія — близько 800.

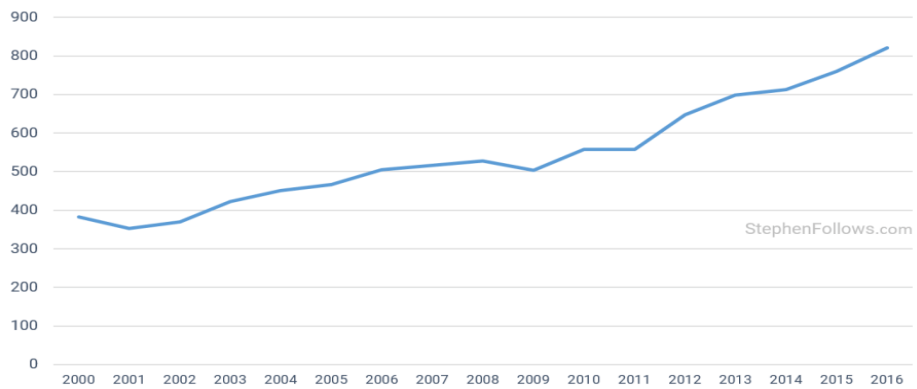

Рисунок 1.2 – Кількість фільмів, випущених в Великобританії, 2000 - 2016

А лідером по продуктивності в цій сфері безумовно є Індія. Боллівуд класифікується як найбільший виробник фільмів у світі з точки зору кількості зайнятих людей та кількості вироблених фільмів. Лише за 2011 рік по всьому світу було продано понад 3,5 мільярда квитків. В Голівуді за цей рік було продано 900 тисяч квитків. Щороку там знімається близько 1041 фільму. В Китаї щороку знімається близько 650 фільмів, в Японії — 600, в Франції — 300. Загалом, кількість стрічок, випущених щороку вже можна рахувати тисячами.

Простому користувачеві, щоб вибрати серед всієї цієї гори матеріалу треба витратити немало часу, чого він робити не збирається. Саме тому сервіси, що надають юзерам можливість обрати кіно, потрібно якось оптимізувати алгоритм видачі. Більшість з них сходяться на думці, що персоналізація здатна покращити досвід користувача і збільшити доходи за рахунок підвищення лояльності до бренду і, як наслідок, збільшити кількість переглядів. Такі підходи базуються на єдиній архітектурі, як базується на машинних алгортмах і налаштуваннях користувача.

1.2 Аналіз існуючих рішень

Математичні методи, які застосовуються при розробці рекомендаційних систем, можна розбити на дві групи: методи коллаборативной фільтрації (collaborative filtering) і тематичні методи (content-based, information filtering). Звичайно, можливе одночасне використання методів двох груп (hybrid prediction). Зазвичай користь від товару представлена певною кількістю балів, що демонструють, наскільки конкретному користувачеві сподобався конкретний товар. У найзагальнішому вигляді проблема зводиться до присвоєння тієї чи іншої оцінки товару, ще не відомого покупцеві. Очевидно, така оцінка дається виходячи з аналізу попередніх переваг даного покупця або будь-якої інший інформації про нього. Після того як система передбачає оцінки для ще не відомих споживачеві товарів: ті з них, які отримують найвищі оцінки рекомендуються потенційному споживачеві.

1.2.1 Коллаборативна фільтрація

Коллаборативна фільтрація аналізує природу кожного елемента. Наприклад, рекомендує певного письменника користувачеві, виконуючи обробку попередніх оцінок його творчості користувачами. З іншого боку, коллаборативна фільтрація не вимагає жодної інформації про об’єкти рекомендацій або користувачів. Він рекомендує елементи на основі їх попередньої поведінки. Існує дві категорії такого типу фільтрації:

- заснована на даних користувачів. Вимірювати схожість між цільовими користувачами та іншими.
- заснована на елементах рекомендацій. Порівнює елементи, з якими взаємодіє юзер з іншими елементами.

Ключовою ідеєю коллаборативної фільтрації полягає в тому, що схожу користувачі мають спільні інтереси і що подібні елементи сподобаються користувачеві.

Припустимо, що є m користувачів та n елементів, ми використовуємо матрицю з розміром m * n для позначення минулого поведінки користувачів. Кожна клітинка в матриці представляє пов'язану оцінку користувача для кожного елементу. Наприклад, M_ {i, j} позначає, як користувач і оцінив елемент j. Така матриця називається матрицею корисності. Якщо певний користувач не оцінював елемент, то у відповідній клітинці буде значення 0. Існує два види оцінювання користувачем: явне та неявне. Перший прямо показує оцінку користувачів (як рейтинг програми або кіно), а другий надає нам евристику про те, як користувач відноситься до елементу (наприклад, кількість лайків, кліків, відвідувань). Явне оцінювання більш прямолінійе, ніж неявне, тому що нам не потрібно вгадувати, що означає це число. Наприклад, там може бути пісня, яку користувач дуже любить, але він прослухав її тільки один раз, тому що був зайнятий під час прослуховування і не зміг її явно оцінити. Без чіткої оцінки, ми не можемо бути впевнені, чи подобається користувачеві цей елемент чи ні. Проте більшість відгуків, які збираються з користувачів, — неявні.

1.2.1.1 Коллаборативна фільтрація, заснована на даних користувачів

Ми знаємо, що нам потрібно обчислити подібність між користувачами в цьому типі фільтрації. Але як ми вимірюємо її? Існує два варіанти: співвідношення Пірсона або косинусна подібність. Нехай u_ {i, k} позначає подібність між користувачем i та користувачем k, а v_ {i, j} позначає оцінку, яку користувач дає елементу j з v_ {i, j} =? якщо користувач не оцінив цей товар. Ці два методи можна виразити наступним чином, за формулами (1.1) та (1.2) :

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \tag{1.1}$$

$$\cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}} \tag{1.2}$$

Обидва ці заходи зазвичай використовуються. Різниця полягає в тому, що кореляція Пірсона незмінна до додавання константих значень до всіх елементів.

Тепер ми можемо прогнозувати думку користувачів щодо предметів без рейтингу за допомогою рівняння (1.3):

$$v_{ij}^{*} = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj} \tag{1.3}$$

Давайте проілюструємо це конкретним прикладом. В наступній матриці кожен рядок відповідає користувачам, а стовпець — різним фільмам, окрім останнього, який фіксує подібність між цим користувачем і цільовим користувачем. Кожна комірка представляє рейтинг, який користувач надає цьому фільму. Припустимо, що користувач E є цільовим.

Таблиця 1.1 – Таблиця оцінювання фільмів користувачами

	Месники	Шерлок	Трансфор-мери	Матриця	Титанік	Спліт	Подібність (i, E)
A	2		2	4	5		не визначено
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			не визначено

Оскільки користувач A та F не мають спільних рейтингів з користувачем E, їх схожість з користувачем E не визначається в кореляції Пірсона. Тому нам потрібно лише розглянути користувачаі B, C і D. Виходячи з кореляції Пірсона, ми можемо обчислити таку схожість.

Таблиця 1.2 – Оцінювання фільмів користувачами з визначеною подібністю між користувачами на основі кореляції Пірсона.

	Месники	Шерлок	Трансфор-мери	Матриця	Титанік	Спліт	Подібність(i, E)
A	2		2	4	5		не визначено
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			не визначено

З наведеної вище таблиці ви можете побачити, що користувач D сильно відрізняється від користувача E, оскільки співвідношення Пірсона між ними є негативним. Він оцінив "Спліт" вище його середнього рейтингу, а користувач "E" зробив протилежне. Тепер ми можемо почати заповнювати пробіл для фільмів, які користувач E не оцінив на основі інших користувачів.

Таблиця 1.3 – Оцінювання фільмів користувачами з розрахованими коефіцієнтами.

	Месники	Шерлок	Трансфор-мери	Матриця	Титанік	Спліт	Подібність(i, E)
A	2		2	4	5		не визначено
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E	3.51*	3.81*	4	2.42*	2.48*	2	1
F	4	5		1			не визначено

Незважаючи на те, що коллаборативна фільтрація, заснована на користувачах дуже проста, вона має декілька проблем. Одне з основних питань полягає в тому, що переваги користувачів можуть з часом змінюватися. Це вказує на те, що попереднє обчислення матриці залежно від сусідніх користувачів може призвести до поганої продуктивності. Щоб вирішити цю проблему, ми можемо самостійно застосувати також фільтрації, які засновані на цих самих елементах.

1.2.1.2 Коллаборативна фільтрація, заснована на елементах рекомендацій

Замість того, щоб вимірювати подібність між користувачами, даний вид фільтрації рекомендує елементи, виходячи з їх подібності з предметами, які цільовий користувач оцінив. Аналогічно, подібність можна обчислити за допомогою кореляції Пірсона або подібності косинуса. Основна відмінність полягає в тому, що при спільній фільтрації з елементами ми заповнюємо порожні клітинки вертикально. У наведеній нижче таблиці показано, як це зробити для фільму “Спліт”.

Таблиця 1.4 – Визначення коефіцієнтів рекомендацій заснованих на елементах рекомендацій

	Месники	Шерлок	Трансформери	Матриця	Титанік	Спліт
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Подіб-ність	-1	-1	0.86	1	1	

Він успішно уникає проблеми, спричиненої змінами переваг користувачів, оскільки фільтрація на основі елементів більш статична. Проте і для цього методу існує кілька проблем. По-перше, головним питанням є масштабованість.

Кількість необхідних розрахунків зростає як з ростом клієнтів, так і з ростом продуктів. Найзатратнішою є складність O (mn) з m користувачами та n елементами. Окрім того, ще одна проблема - ранжування. Подивіться на цю таблицю знову. Хоча існує лише один користувач, який оцінював як “Матрицю”, так і “Титанік”, подібність між ними становить 1. У крайніх випадках ми можемо мати мільйони користувачів, і подібність між двома досить різними фільмами може бути дуже високою просто тому, що вони мають аналогічне значення для єдиного користувача, який оцінив їх обох.

1.2.2 Тематичні рекомендаційні системи

Прогнозування зазвичай проводиться: 1) оптимізацією функції корисності, і емпіричним обґрунтуванням її поведінки, або 2) знаходженням функції корисності, що оптимізує задані параметри поведінки, такі як середньоквадратичне відхилення. Споживач повинен отримати в якості рекомендації продукти з найвищими оцінками з аналізованих. Системи можуть виробляти безліч товарів, найбільш відповідних споживачеві, або сукупність споживачів, найбільш відповідних товару.

Тематичні рекомендаційні системи намагаються знайти схожість між товарами, оціненими людьми раніше. І тільки продукти, що володіють високим ступенем спільності з уподобаннями споживача, будуть рекомендовані. Сучасні підходи до отримання інформації вимагають створення профілів споживачів з смаками, уподобаннями, потребами і т.д. Інформація для профілів може бути отримана від споживача безпосередньо, наприклад, через анкети, або побічно - шляхом аналізу скоєних даними споживачем дій.

Крім традиційних евристичних методів, що ґрунтуються на принципах пошуку інформації, існують інші тематичні методи, такі як класифікатор Байєса, методи машинного навчання, які включають кластеризацію, дерева рішень, штучні нейронні мережі. Ці методи відрізняються тим, що вони пророкують корисність, базуючись не на евристичних алгоритмах, таких як коефіцієнт

лінійного подібності, а на попередніх даних, отриманих шляхом статистичного аналізу і машинного навчання.

Обмеження тематичних методик пов'язані з об'єктами рекомендацій. Тому для адекватної роботи системи потрібна форма, доступна для автоматичного машинного аналізу, або призначення вручну всіх параметрів.

Інша проблема полягає в тому що два різних предмета, з однаковим набором властивостей, невиразні. Хоча фільми зазвичай представлені найбільш репрезентативними ключовими словами, тематичні системи не здатні відрізнити хороший фільм від поганого, хоча вони і використовують одні і ті ж слова.

Також тематичні рекомендаційні системи мислять занадто вузько. Користувач отримує рекомендації тільки таких товарів, які схожі з товарами, які раніше вже отримали його оцінку. Для вирішення цієї проблеми, часто використовується випадковий вибір. У деяких випадках необхідно уникати рекомендацій фільмів занадто схожих на вже відомі, наприклад ремейк того самого фільму.

Рисунок 1.3 – Приклад “вузького мислення” рекомендаційних систем

Як і у всіх рекомендаційних системах, присутній проблема "холодного старту". Користувачеві необхідно оцінити досить велику кількість різних товарів, перш ніж система зуміє правильно зрозуміти його переваги і дати йому відповідні рекомендації. Тому система не зможе давати точні рекомендації новому споживачеві, який ще не виставив багато оцінок.

У будь-якой рекомендаційної системи кількість оцінок, які необхідно передбачити, зазвичай набагато перевищує кількість даних оцінок. Важливо,

щоб система вміла ефективно передбачити оцінки, виходячи з невеликої кількості прикладів. Також необхідна наявність критичної кількості користувачів. В рекомендаційних системах, що займаються кінофільмами, велика кількість фільмів може отримати оцінки лише незначної кількості користувачів, і тоді ці фільми будуть рекомендуватися дуже рідко, навіть якщо оцінки цих небагатьох користувачів були високі. Мала кількість рекомендацій може бути зроблено власникам незвичайних смаків в порівнянні зі смаком більшості, для яких в системі не знайдеться схожих користувачів. Подолати проблему розрідженості оцінок можна, якщо при пошуку схожих користувачів використовувати інформацію про користувача, що міститься в його профілі. Велика частина рекомендаційних методик ґрунтується на обмеженому розумінні користувачів і товарів, аналіз яких в основному обмежується інформацією, що міститься в профілях. За межею аналізу залишаються відомості, що містяться в записах про транзакції користувача і інша доступна інформація. Наприклад, традиційні коллаборативні алгоритми зовсім не використовують інформацію з профілів користувачів і товарів, а обмежуються лише інформацією про зроблені оцінки. Самі профілі все ще залишаються занадто примітивними.

Більшість рекомендаційних систем вимагають від користувача досить активної участі. Наприклад, перш ніж видати рекомендацію про фільм, системі необхідно отримати оцінки великої кількості раніше переглянутих стрічок. Оскільки такий спосіб отримання інформації є не дуже зручним для користувача, створюються методи непрямої оцінки. Наприклад, користувач не оцінив фільм, але переглянув його декілька разів. Однак непрямі оцінки часто страждають неточністю. Таким чином, проблема зниження нав'язливості оцінок при збереженні високої якості рекомендацій досить гостро стоїть перед розробниками. Зокрема, необхідно зрозуміти, яку мінімальну кількість оцінок потрібно отримати від користувача, щоб цього було достатньо для вироблення точних рекомендацій.

До того ж, нинішні рекомендаційні системи оперують в двовимірному просторі Користувач-Товар. Це означає, що вони видають рекомендації,

ґрунтуючись виключно на інформації про користувача або про товар і обходять стороною контекстуальную інформацію, яка може виявитися першочергово важливою в деяких випадках (і при деяких спеціальних обставин). Наприклад, у багатьох випадках, корисність товару або послуги може залежати від того, коли відбувається споживання (пору року, день тижня, час доби). Корисність може також залежати від того, з ким, в якій компанії, за яких обставин споживається продукт.

У таких випадках, проста рекомендація продукту клієнту недостатня; при виробленні рекомендації система повинна звернутися до додаткової контекстуальної інформації про час і обставини передбачуваного споживання. На додаток до традиційних методик побудови споживчого профілю (таким, як опора на ключові слова і анкетну демографічну інформацію) останнім часом з'явилися нові методики, які спираються на аналіз мережевої поведінки і т.д. Ці методики дозволяють врахувати інтереси і переваги користувачів і тим самим розширити користувальницький профіль.

У літературі пропонувалися й інші засновані на різних моделях підходи коллаборативной фільтрації. Інші моделі коллаборативної фільтрації включають аналіз Байєса, вірогідну релятивістську модель, модель лінійної регресії, модель максимальної ентропії. Нещодавно велика кількість робіт було присвячено пошукам більш складних ймовірнісних моделей коллаборатівной фільтрації.

1.3 Теорія нейронних мереж

Нейронні мережі виникли під час досліджень в області штучного інтелекту, а саме, зі спроб відтворити здатність біологічних нервових систем навчатися і виправляти помилки, моделюючи структуру мозку на низькому рівні. Основною областю досліджень з штучного інтелекту в 60-ті - 80-ті роки були експертні системи. Такі системи грунтувалися на високорівневому моделюванні процесу мислення (зокрема, на представленні, що процес нашого мислення побудований на маніпуляціях з символами). Скоро стало зрозуміло, що подібні

системи, хоч і можуть принести користь в деяких областях, не проникають у деякі ключові аспекти людського інтелекту. Згідно з однією з точок зору, причина цього полягає в тому, що вони не в змозі відтворити структуру мозку. Щоб створити штучних інтелект, необхідно побудувати систему зі схожою архітектурою.

Мозок складається з дуже великого числа (приблизно 10,000,000,000) нейронів, з'єднаних численними зв'язками (в середньому кілька тисяч зв'язків на один нейрон, проте це число може сильно коливатися). Нейрони - це спеціальна клітини, здатні поширювати електрохімічні сигнали. При активації нейрон посилає електрохімічний сигнал. Цей сигнал досягає інших нейронів, які можуть в свою чергу активуватися. Нейрон активується тоді, коли сумарний рівень сигналів, які прийшли в його ядро, перевищить певний рівень (поріг активації).

Таким чином, будучи побудований з дуже великого числа зовсім простих елементів (кожен з яких бере зважену суму вхідних сигналів і в разі, якщо сумарний вхід перевищує певний рівень, передає далі двійковий сигнал), мозок здатний вирішувати надзвичайно складні завдання. Зрозуміло, ми не торкнулися тут багатьох складних аспектів будови мозку, однак цікаво те, що штучні нейронні мережі здатні досягти чудових результатів, використовуючи модель, яка не набагато складніше, ніж описана вище.

1.3.1 Базова штучна модель

Щоб відобразити суть біологічних нейронних систем, визначення штучного нейрона дається наступним чином: він отримує вхідні сигнали (вихідні дані або вихідні сигнали інших нейронів нейронної мережі) через кілька вхідних каналів. Кожен вхідний сигнал проходить через з'єднання, що має певну інтенсивність (або вага); ця вага відповідає активності біологічного нейрона. З кожним нейроном пов'язане значне порогове значення. Обчислюється зважена сума входів, з неї віднімається граничне значення і в результаті виходить величина активації нейрона.

Сигнал активації перетворюється за допомогою функції активації (або передавальної функції) і в результаті виходить вихідний сигнал нейрона.

Якщо при цьому використовувати ступінчасту функцію активації (тобто, вихід нейрона дорівнює нулю, якщо вхід негативний, і одиниці, якщо вхід нульовий або позитивний), то такий нейрон буде працювати точно так само, як описаний вище природний нейрон. Насправді, як ми скоро побачимо, порогові функції рідко використовуються в штучних нейронних мережах. Врахуйте, що ваги можуть бути негативними, - це значить, що на нейрон надається не збуджуючий, а гальмівний вплив (в мозку присутні гальмуючі нейрони).

Це був опис окремого нейрона. Тепер виникає питання: як поєднувати нейрони один з одним? Якщо мережу передбачається для чогось використовувати, то у неї повинні бути входи (що приймають значення із зовнішнього світу, які цікавлять нас) і виходи (прогнози або керуючі сигнали). Входи і виходи відповідають сенсорним і руховим нервам - наприклад, відповідно, йде від очей і в руки. Крім цього, однак, в мережі може бути ще багато проміжних (прихованих) нейронів, що виконують внутрішні функції. Вхідні, приховані і вихідні нейрони повинні бути пов'язані між собою.

Ключове питання тут - зворотній зв'язок. Найпростіша мережа має структуру прямої передачі сигналу: сигнали проходять від входів через приховані елементи і в кінці кінців приходять на вихідні елементи. Така структура має стійку поведінку. Якщо ж мережа рекуррентная (тобто містить зв'язки, які проводять сигнали назад від дальніх до ближніх нейронів), то вона може бути нестійка і мати дуже складну динаміку поведінки. Рекурентні мережі становлять великий інтерес для дослідників в області нейронних мереж, однак при вирішенні практичних завдань, принаймні досі, найбільш корисними виявилися структури прямої передачі.

Типовий приклад мережі з прямою передачею сигналу показаний на малюнку. Нейрони регулярним чином організовані в шари. Вхідний шар служить просто для введення значень вхідних змінних. Кожен з прихованих і вихідних нейронів з'єднаний з усіма елементами попереднього шару. Можна було б

розглядати мережі, в яких нейрони пов'язані тільки з деякими з нейронів попереднього шару; однак, для більшості додатків мережі з повною системою зв'язків краще.

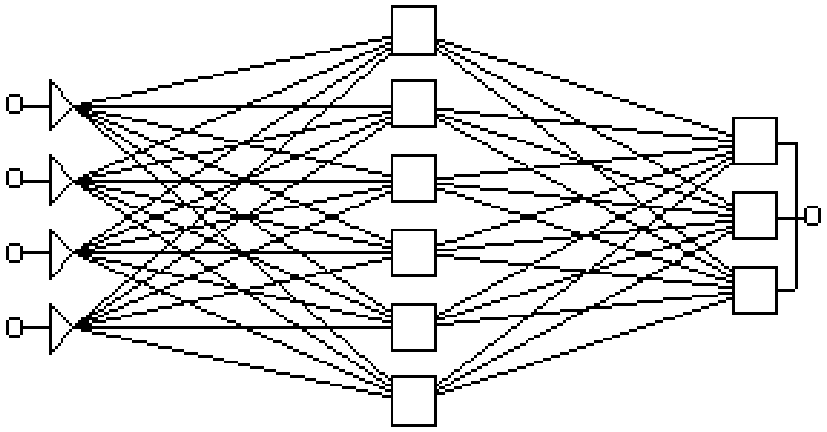

Рисунок 1.4 – Графічна модель нейронної мережі

При роботі мережі у вхідні елементи подаються значення вхідних змінних, потім послідовно відпрацьовують нейрони проміжних і вихідного шарів. Кожен з них обчислює своє значення активації, беручи зважену суму виходів елементів попереднього шару і віднімаючи з неї порогове значення. Потім значення активації перетворюються за допомогою функції активації, і в результаті виходить кінцевий результат. Після того, як вся мережа відпрацює, вихідні значення елементів вихідного шару приймаються за вихід всієї мережі в цілому.

1.3.2 Застосування нейронних мереж

У попередньому розділі в дещо спрощеному вигляді було описано, як нейронна мережа перетворює вхідні сигнали у вихідні. Тепер виникає наступне важливе питання: як застосувати нейронну мережу для вирішення конкретного завдання?

Клас задач, які можна вирішити за допомогою нейронної мережі,

визначається тим, як мережа працює і тим, як вона навчається. При роботі нейронна мережа приймає значення вхідних змінних і видає значення вихідних змінних. Таким чином, мережу можна застосовувати в ситуації, коли у вас є певна відома інформація, і ви хочете з неї отримати деяку поки не відому інформацію. Ось деякі приклади таких завдань:

- Прогнозування на фондовому ринку. Знаючи ціни акцій за останній тиждень і сьогоднішнє значення індексу FTSE, спрогнозувати завтрашню ціну акцій.

- Надання кредиту. Потрібно визначити, чи високий ризик надання кредиту приватній особі, яка звернулася з таким проханням. В результаті розмови з ним відомий його дохід, попередня кредитна історія і т.д.

- Управління. Потрібно визначити що повинен робити робот (повернутися направо або наліво, рухатися вперед і т.д.), щоб досягти мети; відоме зображення, яке передає встановлена на роботі відеокамера.

Зрозуміло, зовсім не будь-яке завдання можна вирішити за допомогою нейронної мережі. Якщо ви хочете визначити результати лотереї, тираж якої відбудеться через тиждень, знаючи свій розмір взуття, то навряд чи це вийде, оскільки ці речі не пов'язані один з одним. Насправді, якщо тираж проводиться чесно, то не існує такої інформації, на підставі якої можна було б передбачити результат.

Отже, ми приходимо до другого істотної умови застосування нейронних мереж: ви повинні знати (або хоча б мати серйозні підозри), що між відомими вхідними значеннями і невідомими виходами є зв'язок. Цей зв'язок може бути спотворений шумом (так, навряд чи можна очікувати, що за даними з прикладу з прогнозуванням цін акцій можна побудувати абсолютно точний прогноз, оскільки на ціну впливають і інші фактори, які не представлені у вхідному наборі даних, і крім того в завданні присутній елемент випадковості), але вона повинна існувати.

Як правило, нейронна мережа використовується тоді, коли не відомий точний вид зв'язків між входами і виходами, - якби він був відомий, то зв'язок

можна було б моделювати безпосередньо. Інша суттєва особливість нейронних мереж полягає в тому, що залежність між входом і виходом знаходиться в процесі навчання мережі. Для навчання нейронних мереж застосовуються алгоритми двох типів (різні типи мереж використовують різні типи навчання): кероване ("навчання з учителем") і не кероване ("без вчителя"). Найчастіше застосовується навчання з учителем.

Для керованого навчання мережі користувач повинен підготувати набір навчальних даних. Ці дані представляють собою приклади вхідних даних і відповідних їм виходів. Мережа навчається встановлювати зв'язок між першими і другими. Зазвичай навчальні дані беруться з історичних відомостей. У розглянутих вище прикладах це можуть бути попередні значення цін акцій і індексу FTSE, відомості про минулих позичальників - їх анкетні дані і те, успішно вони виконали свої зобов'язання, приклади положень робота і його правильної реакції.

Потім нейронна мережа навчається за допомогою того чи іншого алгоритму керованого навчання (найбільш відомим з них є метод зворотного поширення, запропонований в роботі Rumelhart et al., 1986), при якому наявні дані використовуються для коригування ваг і порогових значень мережі таким чином, щоб мінімізувати помилку прогнозу на навчальній множині. Якщо мережа навчена добре, вона набуває здатності моделювати (невідому) функцію, яка б пов'язала значення вхідних і вихідних змінних, і згодом таку мережу можна використовувати для прогнозування в ситуації, коли вихідні значення невідомі.

1.3.3 Збір даних для нейронної мережі

Якщо завдання буде вирішуватися за допомогою нейронної мережі, то необхідно зібрати дані для навчання. Навчальний набір даних являє собою набір спостережень, для яких вказані значення вхідних і вихідних змінних. Перше питання, яке потрібно вирішити, - які змінні використовувати і скільки (і яких) спостережень зібрати.

Вибір змінних (принаймні початковий) здійснюється інтуїтивно. Ваш досвід роботи в даній галузі допоможе визначити, які змінні є важливими.

Нейронні мережі можуть працювати з числовими даними, що лежать в певному обмеженому діапазоні. Це створює проблеми у випадках, коли дані мають нестандартний масштаб, коли в них є пропущені значення, і коли дані є нечисловими. Числові дані масштабуються в відповідний для мережі діапазон, а пропущені значення можна замінити на середнє значення (або на іншу статистику) цієї змінної по всім наявних навчальних прикладів.

Більш важким завданням є робота з даними нечислового характеру. Найчастіше нечислові дані бувають представлені у вигляді номінальних змінних типу Стать = {Чоловік, Жінки}. Змінні з номінальними значеннями можна уявити в числовому вигляді. Наприклад, чоловіча стать — 1, жіноча — 0. Однак, нейронні мережі не дають хороших результатів при роботі з номінальними змінними, які можуть приймати багато різних значень.

Нехай, наприклад, ми хочемо навчити нейронну мережу оцінювати вартість об'єктів нерухомості. Ціна будинку дуже сильно залежить від того, в якому районі міста він розташований. Місто може бути поділені на кілька десятків районів, що мають власні назви, і здається природним ввести для позначення району змінну з номінальними значеннями. На жаль, в цьому випадку навчити нейронну мережу буде дуже важко, і замість цього краще привласнити кожному району визначений рейтинг (грунтуючись на експертних оцінках).

Нечислові дані інших типів можна або перетворити в числову форму, або оголосити незначущими. Значення дат і часу, якщо вони потрібні, можна перетворити в числові, віднімаючи з них початкову дату (час). Позначення грошових сум перетворити зовсім нескладно. З довільними текстовими полями (наприклад, прізвищами людей) працювати не можна і їх потрібно зробити незначущими.

Питання про те, скільки спостережень потрібно мати для навчання мережі, часто виявляється непростим. Відомий ряд евристичних правил, що погоджує

число необхідних спостережень з розмірами мережі (найпростіше з них говорить, що число спостережень має бути в десять разів більше числа зв'язків в мережі). Насправді це число залежить також від (заздалегідь невідомої) складності того відображення, яке нейронна мережа прагне відтворити. З ростом кількості змінних кількість необхідних спостережень зростає нелінійно, так що вже при досить невеликому (наприклад, п'ятдесят) числі змінних може знадобитися величезне число спостережень. Ця трудність відома як "прокляття розмірності".

Для більшості реальних задач буває досить декількох сотень або тисяч спостережень. Для особливо складних завдань може знадобитися ще більшf кількість, проте дуже рідко може зустрітися (навіть тривіальна) завдання, де вистачило б менше сотні спостережень. Якщо даних менше, ніж тут сказано, то насправді у Вас недостатньо інформації для навчання мережі, і краще, що Ви можете зробити - це спробувати підігнати до даних деяку лінійну модель.

У багатьох реальних задачах доводиться мати справу з не цілком достовірними даними. Значення деяких змінних можуть бути спотворені шумом або частково відсутні. Взагалі нейронні мережі в цілому стійкі до шумів. Однак у цій стійкості є межа. Наприклад, викиди, тобто значення, що лежать дуже далеко від області нормальних значень деякої змінної, можуть спотворити результат навчання. У таких випадках краще всього постаратися виявити і видалити ці викиди (або видаливши відповідні спостереження, або перетворивши викиди в пропущені значення).

1.4 Постановка задачі

Оглянувши всі вищеописані методи реалізації задачі, було сформовано бачення функціоналу сайту, що розробляється. Основною його задачею є розроблка алгоритму рекомендацій на базі коллаборативної фільтрації.

Перш за все на даному сайті повинна бути реалізована реєстрація та аутентифікація користувача. Реєстрація представляє собою заповнення форми,

яка складається з набору текстових полів, які потрібно заповнити: нікнейм на сайті, електронна адреса, секретний пароль, його підтвердження та набір тегів. Після заповнення, користувачу на зазначену електронну адресу повинен прийти лист із підтвердженням реєстрації. Після цього користувачу надається доступ до всіх можливостей сервісу. Процес аутентифікації схожий та простіший. Повинна бути можливість зайти на сайт під власним аккаунтом. Для цього повинна бути форма із заповненням нікнейму та паролю.

Після реєстрації чи аутентифікації юзер буде перенаправлений на головну сторінку сайту, де він зможе побачити список всих фільмів, доступних в системі. Користувач зможе оцінювати кінострічку по 10-бальній шкалі, переглядати коментарі інших юзерів і залишати власні, а також переглянути список фільмів, реконмендованих спеціально для нього.

Даний функціонал планується в майбутньому розширити, додавши в систему функціонал для оцінювання емоціонального забарвлення коментарів, що залишають користувачі. Потім модель нейронної мережі зможе враховувати і ці коментарі при побудові рекомендацій.

2 ВИБІР ЗАСОБІВ РОЗРОБКИ ВЕБ-САЙТУ ТА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1 Опис сайту

Веб-сайт, що буде розроблено, матиме можливість реєстрації та авторизації. Він буде призначений для відображення користувачеві списку кінострічок, випущених з 1970 року й можливості видавати йому список фільмів, що найбільше будуть відповідати його перевагам. Також користувач матиме змогу оцінити фільм й залишити коментар до нього.

Інформаційна система повинна:

- використовувати всі останні технічні досягнення в технологіях, що використовуються;
- використовувати клієнт-серверну модель;
- бути багатокористувальницькою системою;
- використовувати єдину базу данних;
- гарантувати клієнту захищеність його персональних даних.

2.2 Вибір програмного забезпечення для вирішення поставленого завдання

В роботі використовується архітектура клієнт-сервер, яка включає в себе такі компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

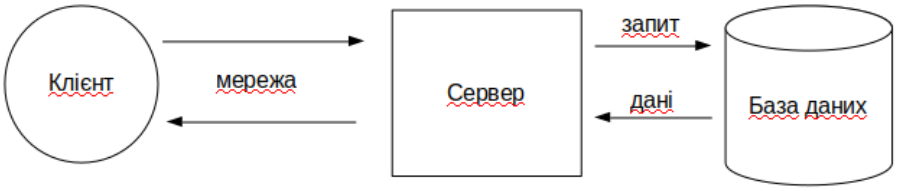

Рисунок 2.1 – Архітектура клієнт-сервер

Для розробки клієнтської та серверної частини пропонується безліч бібліотек, мов програмування, фреймворків та технологій і вибір між ними може стати досить складною задачею.

2.2.1 Вибір стеку технологій для клієнтської частини

Для клієнтської частини використовується HTML, CSS та, звичайно, мова JavaScript. Нещодавно JavaScript зайняв місце серед кращих мов для вивчення за версією IBM в 2017 році. На цьому етапі він використовується як для клієнтської, так і для серверної частини та допомагає проектувати привабливі інтерфейси, збагачувати веб-додатки багатьма функціями, змінювати веб-сторінки в реальному часі та багато іншого.

Тим часом, JavaScript фреймворки можуть стати ключовим інструментом для швидкої розробки веб-сайтів. Вони служать каркасом для окремих додатків, що дозволяють розробникам менше турбуватися про структуру коду або підтримку в той час, коли вони зосереджені на створенні складних елементів інтерфейсу.

Переваги використання JavaScript фреймворків:

- Ефективність - проекти, які раніше зайняли б місяці і сотні рядків коду, зараз можуть бути реалізовані набагато швидше з добре структурованими готовими шаблонами і функціями.

- Безпека - кращі JavaScript фреймворки мають добре опрацьовану систему безпеки і підтримуються великим співтовариством, члени і просто

користувачі якого виступають в ролі тестувальників.

- Витрати - більшість фреймворків з відкритим кодом і безкоштовні. Оскільки вони допомагають програмістам швидше розробляти власні рішення, підсумкова ціна веб-сайту буде нижче.

Для розробки клієнтської логіки було вибрано React.js, не дивлячись на той факт, що це більше бібліотека, ніж фреймворк. Він використовується в інтерфейсах Фейсбук і інстаграмма, показуючи свою ефективність всередині динамічних додатків з високим трафіком (із відповідною пропускною спроможністю).

Він по праву вважається самим швидкозростаючим JS фреймвком: на сьогодні налічується близько 1000 авторів Github. У MVC (Model-View-Controller) моделі React.js діє як "V" і може бути легко інтегрований в будь-яку архітектуру. Завдяки використанню віртуального DOM (document object model) дерева він забезпечує більше зростання продуктивності в порівнянні з Angular 1.x. Замість того, щоб взаємодіяти з DOM безпосередньо, ми працюємо з його швидшою копією. Ми можемо вносити зміни в копію, виходячи з наших потреб, а після цього застосовувати зміни до реального DOM.

При цьому відбувається порівняння DOM-дерева з його віртуальної копією, визначається різниця і запускається повторний рендеринг того, що було змінено.

Такий підхід працює швидше, бо не включає в себе всі великовагові частини реального DOM. Але тільки якщо ми робимо це правильно. Є дві проблеми: коли саме робити повторну перерисовку DOM і як це зробити ефективно. Перше: коли дані змінюються і інтерфейс потребує оновлення. Є два варіанти дізнатися, що дані змінилися. Перший з них - «dirty checking» (брудна перевірка) полягає в тому, щоб опитувати дані через регулярні проміжки часу і рекурсивно перевіряти всі значення в структурі даних. Другий варіант - «observable» (спостережуваний) полягає в спостереженні за зміною стану. Якщо нічого не змінилося, ми нічого не робимо. Якщо змінилося, ми точно знаємо, що потрібно оновити. Друге: що робить цей підхід дійсно швидким:

- ефективні алгоритми порівняння;
- угруповання операцій читання / запису при роботі з DOM;
- ефективне оновлення тільки дочірніх гілок DOM дерева.

На додаток до цього, компоненти React можуть бути створені і повторно використані в інших додатках або навіть передані для громадського використання.

Не дивлячись на той факт, що React більш складна у вивченні, ця бібліотека робить розробку додатків простою і легкою для розуміння. Крім того, він може ідеально підійти для складних, значних програмних рішень з високим ступенем навантаження.

React використовує Flux архітектуру. Це не фреймворк, або бібліотека, це новий архітектурний підхід, який доповнює React і принцип односпрямованого потоку даних.

Типова реалізація архітектури Flux може використовувати цю бібліотеку разом з класом EventEmitter з NodeJS, щоб побудувати подієво-орієнтовану систему, яка допоможе керувати станом додатки.

Ймовірно, Flux найлегше пояснити, виходячи зі складових його компонентів:

- Actions (дії) - хелпери, що спрощують передачу даних Диспатчеру;
- Dispatcher (диспатчер) - приймає Дії і розсилає навантаження зареєстрованим обробникам;
- Stores (сховища) - контейнери для стану програми та бізнес-логіки в обробниках, зареєстрованих в диспетчері;
- Controller Views (відображення) - React-компоненти, які збирають стан сховищ і передають його дочірнім компонентів через властивості.

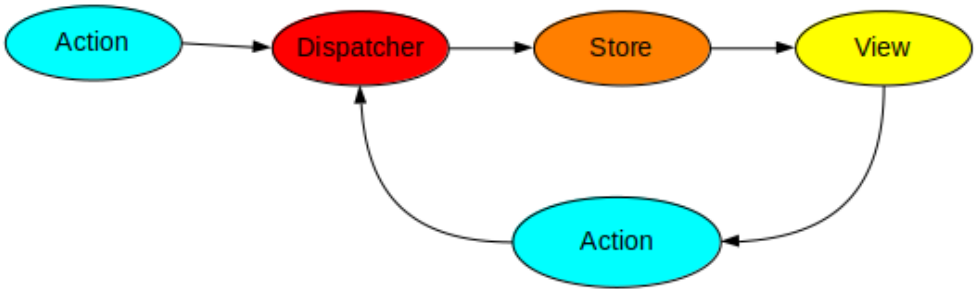

Рисунок 2.2 – Архітектура Flux

Наприклад, користувач тисне на кнопку завантаження списку фільмів. В такому випадку в диспатчер передається відповідний action і виконується http запит до серверу. Клієнт отримує данні і поміщує їх у відповідне сховище. Компоненти(відображення) отримують нові дані зі сховища і відображують їх, попередньо обробивши. Перевагою цих компонентів є їх модульність. Тобто ви можете використовувати компоненти, які були раніше написані вами або кимсь. Наприклад, в роботі я використовував компонент вводу (input), розроблений мною раніше для іншого проекту.

Для реалізації сховища використовується бібліотека Redux, яка слідує основним принципам функціонального програмування. За функціональним програмуванням стоїть декілька ключових ідей.

Іммутабельність. Іммутабельність передбачає, що після створення дані або структура, яка їх містить, не можуть бути змінені. На практиці існує два типи мутацій (змін) - видимі і невидимі. Видимі мутації - це мутації, які змінюють дані або структуру, що містить ці дані, способом, який контролюється зовнішнім спостерігачем через API. Невидимі мутації не можуть контролюватися через API (хорошим прикладом є кешуючі структури даних). У певному сенсі невидимі зміни можуть розглядатися як сайд-ефекти (side-effects) (ми розглянемо цей аспект і його значення трохи пізніше). В контексті функціонального програмування зазвичай заборонені обидва види модифікацій: не тільки дані є

іммутабельнимі за замовчуванням, але і структури даних не можуть бути змінені після створення. Коли розробники (і компілятори або середовища виконання) можуть бути впевнені, що дані не можуть змінитися, з'являються досить цікаві переваги:

- блокування для багатопоточності більше не є проблемою: якщо дані не змінюються, немає необхідності в будь-яких блокуваннях, щоб синхронізувати різні потоки;
- зберігання (persistence) - ще один ключовий концепт, який ми розглянемо пізніше, стає простіше;
- копіювання можливо за константне час, тому що цей процес стає лише питанням створення нового посилання на вже існуючий екземпляр структури даних;
- в деяких випадках порівняння значень може бути оптимізовано: коли середовище виконання або компілятор під час завантаження або компіляції можуть бути впевнені, що одна змінна дорівнює інший тільки в тому випадку, якщо вони обидві вказують на один і той же об'єкт в пам'яті, глибоке порівняння значень стає порівнянням посилань. Цей процес відомий як інтернування і зазвичай можливий тільки для даних, доступних під час компіляції або завантаження.

Сайд-ефекти. В теорії мов програмування сайд-ефекти(ефекти на стороні) будь-якої операції (зазвичай це виклик функції або методу) визначаються як спостережувані ефекти, які можуть бути помітні за межами виклику функції (змінюють стан за межами своєї області видимості). Іншими слова, після виклику ви побачите зміну стану. Кожен виклик мутує (змінює) стан. На відміну від концепції іммутабельності, яка зазвичай пов'язана тільки з даними або структурою даних, сайд-ефекти стосуються стану програми в цілому. Функція, яка зберігає іммутабельность структури даних, може викликати побічні ефекти. Хорошим прикладом цього служать кешуючий функції або мемоізація. Незважаючи на те, що зовнішньому спостерігачеві може здаватися, що ніяких змін не відбулося, оновлення глобального або локального кешу має побічний

ефект поновлення внутрішніх структур даних, які працюють як кеш (результуючі прискорення також є побічним ефектом). Завдання розробника - знати про побічні ефекти і вміти працювати з ними. Наприклад, продовжуючи приклад з кешем, іммутабельна структура даних, що кешує звернення до неї для прискорення роботи, не може безпечно передаватися між потоками: або кеш повинен підтримувати багатопоточність, або це може призвести до несподіваних результатів. Функціональне програмування як парадигма передбачає використання функцій, які не викликають сайд-ефектів. Це означає, що функції повинні просто виконувати операції над даними, переданими в якості параметрів, і ефекти цих операцій повинні бути видні тільки тому, хто викликав цю функцію. Іммутабельні структури даних йдуть рука об руку з функціями без сайд-ефектів.

Чистота. Це ще одне обмеження, яке може бути накладено на функції: чисті функції при обчисленні результату використовують тільки те, що явно передано їм в якості параметрів. Іншими словами, чисті функції не можуть використовувати глобальний стан або стану, доступні через якісь інші конструкції.

2.2.2 Вибір стеку технологій для серверної частини

Якщо на стороні клієнту безумовним лідером є JavaScript, то на стороні серверу не все так просто. Майже всі більш-менш відомі мови зараз надають програмістам функціонал для створення http-серверів, навіть вище згаданий JavaScript.

Для реалізації серверної частини була вибрана мова програмування Python. Python — це мова програмування загального призначення, націлена в першу чергу на підвищення продуктивності самого програміста, ніж коду, який він пише. Говорячи простою людською мовою, на Python можна написати практично що завгодно (веб або настільні додатки, ігри, скрипти по автоматизації, комплексні системи розрахунку, системи управління

життєзабезпеченням і багато іншого) без відчутних проблем. Більш того, поріг входження низький, а код багато в чому лаконічний і зрозумілий навіть того, хто ніколи на ньому не писав. За рахунок простоти коду, подальший супровід програм, написаних на Python, стає легше і приємніше в порівнянні з Java або C++. А з точки зору бізнесу це тягне за собою скорочення витрат і збільшення продуктивності праці співробітників. Серед переваг Python можна відмітити:

- вбудовані структури даних, словники, кортежі;
- простий і зручний синтаксис;
- велика кількість бібліотек;
- потужні інтерфейси до конкретних ОС;
- переносимість коду між платформами: автоматичну генерацію документації для модуля і можливість написання самодокументуючихся програм;
- підтримку процедурного, функціонального і об'єктного стилів програмування; вбудовану підтримку Unicode і велика кількість національних кодувань.
- Мова легко об'єднується з написаними на C і C ++ модулями, що дозволяє істотно збільшити швидкість програм. Завдяки цим особливостям розгортка додатків може виконуватися дуже швидко.
- Python надає нам велику кількість бібліотек для роботи з матрицями, нейромережами та математичними обчисленнями.

Також навколо Python сформувалась одна з найбільших спільнот і це дає нам, звичайним програмістам, ряд значних переваг. Активна спільнота визначає розвиток мови в цілому. Для великих корпорацій це показник надійності з точки зору бізнесу. YouTube, Reddit, Quora, Spotify не бояться робити свої продукти на Python. Не кажучи про невеликі компанії. А значить, поки їх бізнес розвивається, попит на пітоністов продовжить рости.

Також велика спільнота сприяє швидкому навчанню, адже чим більше ком’юніті, тим менше ймовірність, що ваше питання на stackoverflow або github issues залишеться без відповіді. У співтоваристві легко знайти тих, з ким можна

об'єднатися в команду по роботі над проектом.

Для розробки самого веб-серверу було обрано фреймворк Django. Django - це високотехнологічна платформа Python Web, яка сприяє швидкому розвитку та чистому прагматичному дизайну. Побудований досвідченими розробниками, він дбає про велику частину складності веб-розробки, тому ви можете зосередити увагу на написанні вашого додатка без необхідності винаходити колесо. Фреймворк відкритий та безкоштовний, має детально та гарно оформлену документацію.

Кожен раз при розробці веб-сайтів потрібні схожі компоненти: спосіб аутентифицировать користувачів (вхід, вихід, реєстрація), панель управління сайтом, форми, інструменти для завантаження файлів і т. д.

На щастя для нас, інші люди звернули увагу на виникнення однотипних проблем при веб-розробці, так що вони об'єдналися і створили фреймворки (Django і інші), які пропонують нам готові шаблони для використання.

Фреймворк має встроєний ORM (Object-relational mapper). В абсолютній більшості випадків роботи з ORM нам не потрібне використання SQL-синтаксису в виразах, що автоматично знижує ризик появи SQL-injection уразливості.

Однією з унікальних функцій Django є автоматична генерація панелі адміністратора, у якій практично немає аналогів. Крім того, що дана функціональність дозволяє значно скоротити час на написання потрібного інтерфейсу адміністратора, вона ще й дає можливість клієнтам відразу почати працювати з сайтом ще на початкових етапах його розробки. Фактично, досить створити потрібні моделі, і можна відразу показувати сайт клієнту, і вже інтерактивно з ним обговорювати бізнес-логіку, не відволікаючись на дизайн.

Також Django підтримує MTV (Model-Template-View). Даний патерн проектування дуже близький до класичного MVC, і найголовніше, що він дозволяє - це добре відокремлювати бізнес-логіку від дизайну.

Django проект складається з додатків, що являють собою пакет Python, який слідує певній конвенції. Кожен додаток відповідає за конкретний функціонал,

містить свої моделі, функції відображення (View). Наприклад, в даному проекті окремі додатки створені для розробки REST API та роботи з моделями фільмів, відгуків, оцінок і т.д.

Щоб зрозуміти, для чого ж нам потрібен Django, нам потрібно ближче познайомитися з серверами. По-перше, сервер повинен дізнатися про те, що ми чекаємо від нього веб-сторінку.

Коли на сервер приходить запит, він переадресується Django, який намагається уточнити, що конкретно від нього просять. Для початку він бере адрес веб-сторінки і спробує зрозуміти, що потрібно зробити. Ця частина процесу в Django виконує urlresolver. Він не дуже засвоєний, тому просто берет список шаблонів і намагається зв'язати їх з URL-адресою. Django звіряє шаблони згори вниз і, якщо щось збігається, він пересилає запит відповідній функції (яка називається View). В ній ми, наприклад, за допомогою моделей, можемо звертатися до бази даних за певною інформацією, змінювати або фільтрувати її. Модель є єдиним, остаточним джерелом інформації про ваші дані. Він містить основні поля та поведінку даних, які ви зберігаєте. Як правило, кожна модель відображає єдину таблицю бази даних. При створенні об’єкта моделі необхідно створити файл міграції для неї і виконати його. На щастя, Django здатен виконувати це все автоматично, варто лише ввести в терміналі відповідні команди.

2.2.3 Вибір середовища розробки програмного забезпечення

Розробка веб-застосунків мовами Django та JavaScript може здійснюватися як в будь-якому простому текстовому редакторові, наприклад, блокноті Windows.

Також є можлівість реалізації і в потужних WYSIWYG-редакторах або інтегрованих середовищах розробки.

До найбільш відомих редакторів відносяться:

- WebStorm;
- PyCharm;
- Atom;
- Sublime Text 3;
- Visual Studio Code.

Перші два являють собою повноцінні середовища розробки з великою кількістю можливостей, наданих “з коробки”, тобто їх потрібно встановлювати додатково. Решта — текстові редактори з базовим функціоналом. Вони потребують встановлення необхідних нам плагінів з спеціальних магазинів, але, за рахунок цього, займають менше місця і працюють значно швидше. Лідером серед них на сучасному ринку є Visual Studio Code від Microsoft. Його ми і обрали. Його переваги:

- Менеджер для завантаження плагінів вбудований в редактор.
- Вбудований Emmet.
- Більш чіткі вертикальні відступи (важливо для Python).
- Вбудована підтримка Git.

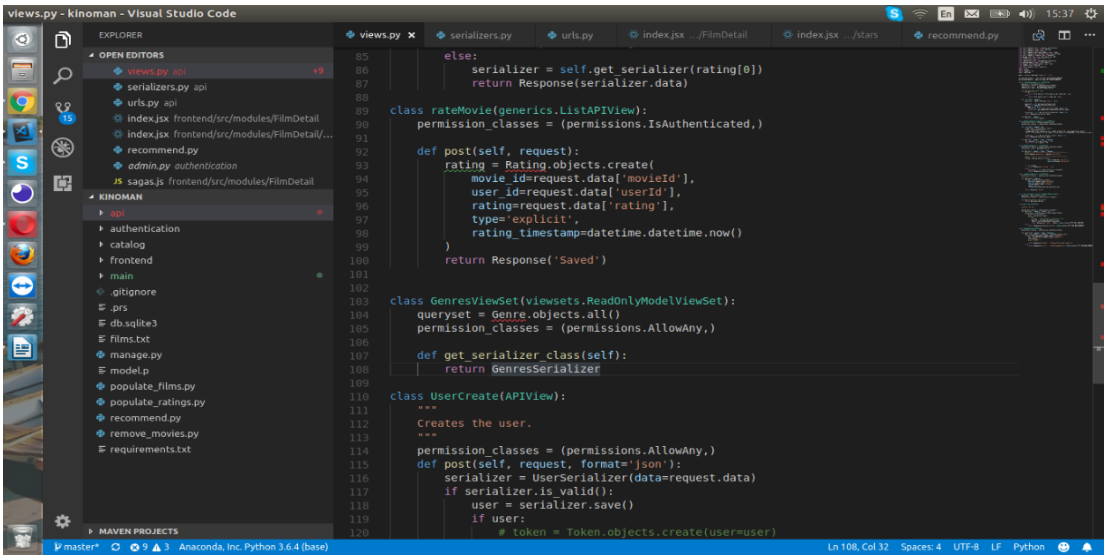

Рисунок 2.3 – Вікно редактора VS Code

2.2.4 Вибір системи управління базами даних

Кожен власник сайту знає, що для правильного функціонування сайту потрібні не тільки файли з кодом сторінок, але і бази даних. Для взаємодії з базами даних використовуються системи управління базами даних (СКБД). Як можна здогадатися вже з назви, система управління базами даних являє собою програмне забезпечення, яке використовується для створення та роботи з базами даних. Головна функція СУБД - це управління даними (які можуть бути як у зовнішній, так і в оперативній пам'яті). СУБД обов'язково підтримує мови баз даних, а також відповідає за копіювання та відновлення даних після будь-яких збоїв.

Реляційні і об'єктно-реляційні СУБД є одними з найпоширеніших систем. Вони являють собою таблиці, у яких кожен стовпець (який називається "field" або "поле") впорядкований і має певне унікальну назву. Для керування базами даних застосовується особлива мова програмування - SQL. Скорочення розшифровується як "Structured query language". Команди, які використовуються в SQL, діляться на ті, які маніпулюють даними, ті, які визначають дані, і ті, які керують даними.

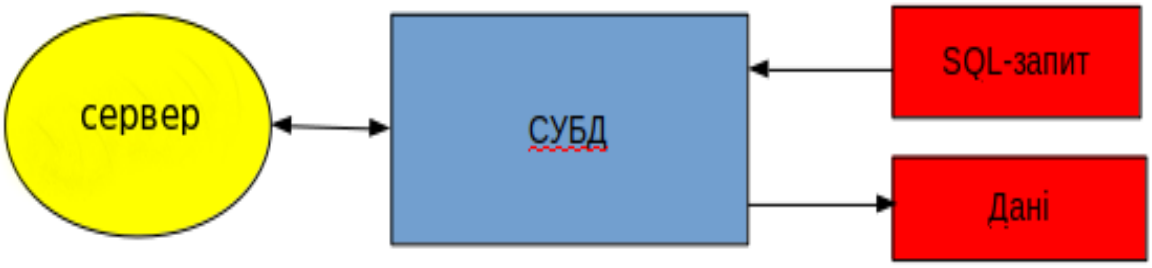

Рисунок 2.4 – Схема взаємодії сервера і СУБД

В данній роботі використовується СУБД postgresql. Ця вільно поширювана система управління базами даних відноситься до об'єктно-реляційному типу СУБД. Як і у випадку з MySQL, робота з PostgreSQL грунтується на мові SQL, однак, на відміну від MySQL, PostgreSQL підтримує стандарт SQL-2011.

Ця СУБД не має обмежень ні щодо максимального розміру бази даних, ні по максимуму записів або індексів в таблиці. Якщо говорити про переваги PostgreSQL, то, безумовно, це надійність транзакцій і реплікації, можливість наслідування і легка розширюваність. PostgreSQL підтримує різні розширення і варіанти мов програмування, такі як PL / Perl, PL / Python і PL / Java. Також є можливість завантажувати C-сумісні модулі.

Багато хто відзначає, що на відміну від MySQL дана СУБД має хорошу і детальну документацію, яка дає відповіді практично на всі питання. Про те, що це більш масштабна, ніж MySQL, СУБД, говорить і той факт, що PostgreSQL періодично порівнюють з такою потужною системою управління даних, як Oracle. Все це дозволяє говорити про PostgreSQL як про одну з найбільш просунутих СУБД на даний момент.

3 РОЗРОБКА БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вимоги до проектування бази даних

Функціональна повнота. Це вимога БД забезпечується обліком інформаційних вимог усіх потенційних користувачів.

Мінімальна надмірність. Надмірність даних є причиною аномалій включення, видалення і редагування даних в БД і, як наслідок, є причиною порушень таких важливих властивостей БД як: цілісність, несуперечність, логічна і фізична незалежність, розширюваність. Мінімальна надмірність досягається виключенням дублюючих елементів цих, обчислюваних елементів, що описують предметну область, і нормалізацію логічного представлення даних.

Цілісність БД. Властивість БД, яка дає можливість зберігати певні обмеження значення даних при усіх модифікаціях БД. У БД розрізняють такі види цілісності : цілісність домена, цілісність таблиці, посилальна цілісність, цілісність.

Цілісність домену – забезпечує приналежність кожного елементу логічного запису певному домену. Ця властивість забезпечується засобами СУБД за допомогою таких параметрів: ім'я поля, тип даних, точність для числових полів, діапазон числових змінних;

Цілісність таблиці – забезпечує унікальність кожного логічного запису в ній. У кожній таблиці дотримується цей тип цілісності за допомогою первинного ключа запису;

Посилальна цілісність – між таблицями даних витримана за допомогою зовнішніх ключів. Між зв'язними таблицями встановлено каскадне оновлення даних і заборону на видалення запису в батьківській таблиці, якщо в дочірній таблиці є хоч би один запис, що містить посилання на запис, що видаляється.

Несуперечність. Одним з найбільш важливих джерел суперечності даних є наявність ненормалізованих стосунків логічної моделі предметної області. Тому

на етапі проектування реалізації необхідно забезпечити наявність в даталогічній моделі стосунків, що тільки не приводяться. Ще одна причина виникнення суперечності даних - це помилки введення даних. В даному випадку несуперечність забезпечується автоматичною перевіркою на дублювання кодів.

Безпека. Властивість безпеки забезпечується доступом до системи тільки через пароль. Доступ до файлів адміністратора БД має тільки адміністратор БД по спеціальному паролю.

Відновлюваність. Забезпечується адміністратором БД шляхом регулярного створення страхових копій файлів даних і збереження їх в страховій директорії. Забезпечується також можливість оновлення файлів БД з резервної директорії у разі виникнення нештатної ситуації.

Узгодженість. В процесі розробки автономного локального варіанту розміщення БД послідовності дій користувачів розмежовується доступом до даних. Довідкова інформація і розклад призначено тільки для перегляду, доступ до інших даних забезпечується відповідною організацією застосунка через меню, кнопки і так далі. Адміністратор БД має можливість увійти до БД у будь-який момент, скориставшись своїм паролем, при цьому поточна робота користувача не припиняється.

Ефективність. Необхідно забезпечити мінімальний час відгуку в процесі виконання кожного застосунку при обмеженнях на займану пам'ять. Ефективність визначається оптимальним набором комплексу апаратно-технічних засобів, операційної системи, СУБД, побудовою оптимальної логічної і фізичної моделі даних в процесі фізичного проектування БД.

Логічна і фізична незалежність. Забезпечується нормалізацією логічного представлення моделі даних предметної області і розробкою не фізичному рівні універсальних програмних модулів.

Розширюваність або відкритість. Забезпечується оптимальною структурою даних в сенсі незалежності логічного і фізичного представлення даних. Це забезпечує незмінність отриманої моделі даних при розширенні меж предметної області, тобто при зміні розробленої структури даних, додаванням

нових структур і зв'язків з існуючими або просто при додаванні нових запитів до БД у вигляді нових оброблювальних програм.

Дружність інтерфейсу користувача. Забезпечується підказками, запитами на підтвердження дій користувача, меню і т.д.

3.2 Вербальний опис моделі інформаційної системи

Кожен фільм має свого режисера і групу акторів. Також всі їх можна поділити на жанри. Деякі можуть належати до декількох жанрів одночасно. Також в системі приведена система рейтингів і коментарів. Як рейтинг, так і коментар відноситься до конкретного користувача системи і конкретної кінострічки.

Сутностями предметної області є:

- Користувач;
- Кінофільм;
- Режисер;
- Актор;
- Рейтинг фільму;
- Коментар;
- Жанр;

3.3 Концептуальна модель інформаційної системи

Концептуальна модель інформаційної системи представляється діаграмою "сутність-зв'язок" (ER-діаграмою).

В системі приведена система рейтингів і коментарів. Як рейтинг, так і коментар відноситься до конкретного користувача системи і конкретної кінострічки.

Тут же показана кратність відношень і властивості (атрибути) сутностей.

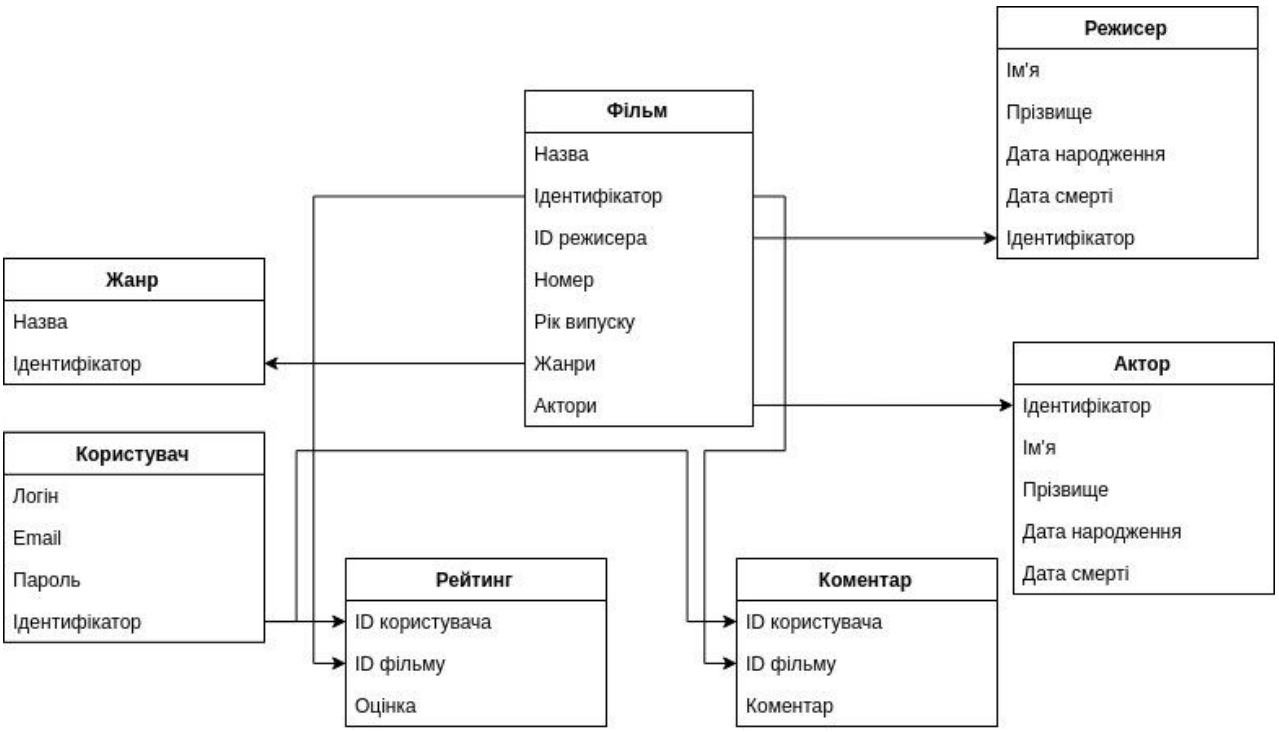

Рисунок 3.1 – ER-діаграма бази даних

Структура концептуальної моделі даних зображена в таблиці. Тут представлені сутності з їх атрибутами і типами полів. У цій таблиці детально описано, які з атрибутів є ключовими полями цієї сутності.

Таблиця 3.1 Таблиця сутностей

Найменування сутності	Опис	Найменування атрибуту	Тип даних	Ключове поле
Фільм	Зберігає інформацію про фільм	Назва	String	ні
		Ідентифікатор	String	так
		ID режисера	Int	ні
		Номер	Int	ні
		Рік випуску	Date	ні

		Жанри	[Int]	ні
		Актори	[Int]	
Режисер	Зберігає інформацію про режисера	Ім’я	String	ні
		Прізвище	String	ні
		Дата народження	Date	ні
		Дата смерті	Date	ні
		Ідентифікатор	Int	так
Актор	Містить інформацію про актора кіно	Ідентифікатор	Int	так
		Ім’я	String	ні
		Прізвище	String	ні
		Дата народження	Date	ні
		Дата смерті	Date	ні
Коментар	Містить інформацію про коментар до фільму та про його автора	ID користувача	Int	ні
		ID фільму	String	ні
		Коментар	String	ні
Оцінка	Оцінка фільму користувачем	ID користувача	Int	ні
		ID фільму	String	ні
		Оцінка	Int	ні
Користувач	Інформація	Логін	String	ні

	про користувача системи	Email	String	ні
		Пароль	String	ні
		Ідентифікатор	Int	так
Жанр	Інформація про жанр фільму	Назва	String	ні
		Ідентифікатор	Int	так

3.4 Вхідна і вихідна інформація системи

Вхідна інформація приведена в табл. 3.2.

Таблиця 3.2 Опис вхідних даних

Назва	Опис
Фільми	Перелік фільмів з 1970 року
Користувачі	Дані авторизації користувачів
Рейтинги	Перелік оцінок кінострічок користувачами

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ РЕКОМЕНДАЦІЙ

4.1 Структурна схема веб-застосунку інформаційної системи

В проекті мається 8 елементів, вартих уваги:

- папка api. Це один з додатків, згенерованих за допомогою Django. Являє собою пакет Python з файлом __init__. Файли __init__.py необхідні, щоб Python розглядав каталоги як ті, що містять пакет; це робиться для того, щоб запобігти ненавмисного приховування потрібного модулю, який Python шукає при його підключенні. Така ситуація може виникнути, якщо десь на шляху пошуку пакету буде лежати папка з такою же назвою, як у потрібного модулю. В папці api міститься логіка для реалізації REST API.
- папка catalog. В ній описані основні моделі системи та деяка логіка для роботи з ними.
- папка main. Головна папка проекту, згенерована разом з проектом. Містить файл settings.py з основними налаштуваннями бази даних, підключеними додатками і т.д.
- папка frontend. Містить всю логіку додатку React.js і клієнтської частини.
- файл populate_film.py. Дані про фільми завантажуються зі стороннього сервісу і зберігаються в базу даних.
- populate_ratings.py. Скрипт завантажує рейтинги до різних фільмів зібраних з твіттера. Перевіряє наявність в базі користувача, якому буде присвоєна оцінка і створює його, якщо він відсутній.
- train_model.py. Один з головних файлів, який тренує нейронну мережу.
- manage.py. Файл, згенерований Django, призначений для запуску серверу і виконанню багатьох інших команд.

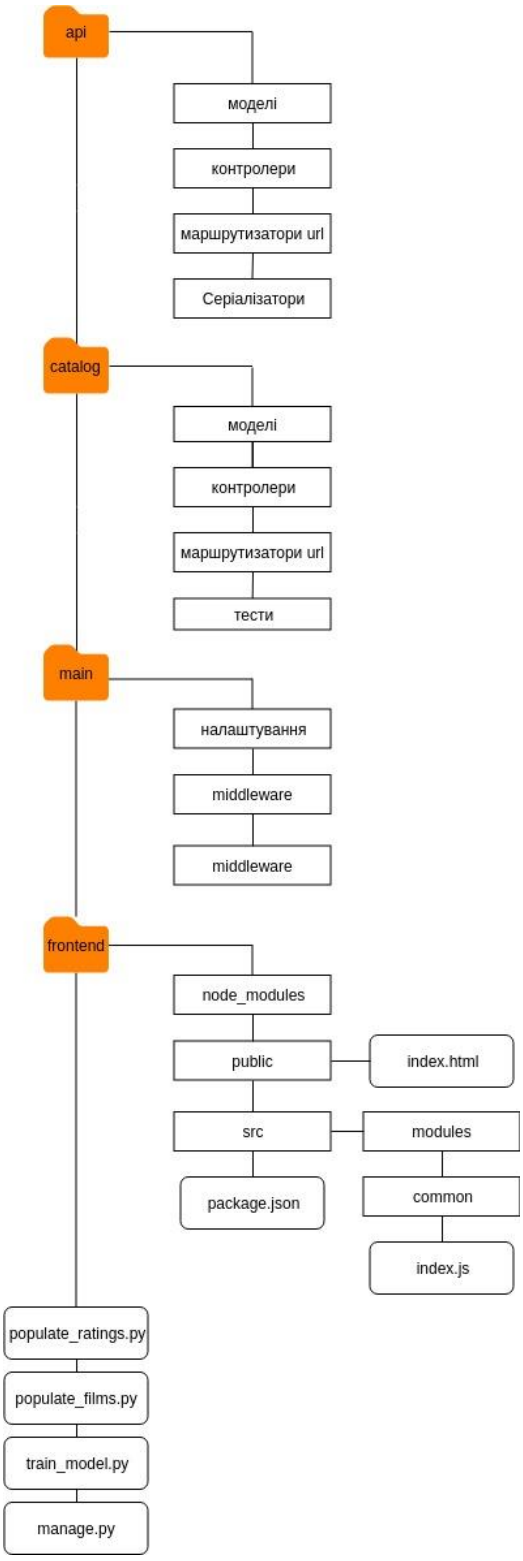

Рисунок 4.1 – Структура папки з проектом

4.2 Програмна архітектура системи, що розробляється

Як вже було вказано вище, головним додатком сайту на Django є папка main. В файлі settings.py вказуються підключені інші додатки, мідлвари, налаштування роботи з базою даних і т. д. (Додаток Б.1.1).

В urls.py в додатку Б.1.2 описаний головний роутинг для шляхів, до яких треба звертатися клієнтському додатку за даними. В кожному додатку є такий файл, який потім підключається в urls.py додатку main. admin — шляхи для адмін панелі, наданої Django. api — безпосередньо шляхи для клієнта.

В папкі catalog описані основні моделі для бази даних і роботи з ними. (Додаток Б.1.3)

В папкі api міститься вся основна логіка серверної частини, а саме контролери в файлі views.py. Саме в цьому файлі завантажується масив з рекомендаціями. Вона зберігається в файл model.p і для її завантажується (як і для сериалізації) використовується бібліотека pickle. Код описаний в додатку Б.1.4

4.3 Розробка алгоритму рекомендацій

В файлі recommends.py описаний алгоритм розробки нейронної мережі для рекомендацій за допомогою бібліотеки tensorflow. Код вказаний в додатку Б.1.5. Також будуть використані такі корисні бібліотеки:

- NumPy для роботи з матриця і математичних обчислень;
- Pandas для аналізу і маніпуляції даними.

Спочатку завантажуємо дані з бази для навчання мережі і згенеруваємо массив.

Далі можемо почати встановлювати параметри нейронної мережі. Спочатку встановимо розмір кожного прихованого шару, в нас їх буде два. X — це просто змінна яка буде використана пізніше. Ваги та нейрони зміщення (biases) — це змінні, ініційовані випадковими змінними типу float.

Далі визначаємо нашу модель. Autoencoders — це навчальні нейронні мережі. Як тільки структура нейронної мережі була визначена, нам потрібна функція втрат. Далі тренуємо нашу модель.

Ми розбиваємо дані для навчання на партії та подаємо їх мережі. Використання такої техніки корисно для прискорення тренувань, оскільки ваги оновлюються при використанні кожної партї.

Ми готуємо нашу модель з векторами рейтингів користувачів, кожен рядок представляє користувача а кожен стовпець — елемент. Записи в клітинках матриці — це оцінки фільмів.

Таблиця 4.1 Розріджена матриця оцінювання

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

Ідентифікатори фільмів і користувачів зберігаються в відповідних таблицях в базі даних(catalog_film, auth_user). Для оцінок створено окрему таблицю, що в рядку містить ідентифікатор фільму, користувача, що його оцінив і власне оцінка(api_rating). Після генерації матриці вона буде використана для навчання моделі передбачень.

Модель буде тренуватися на протязі 100 епох на партіях розміром 250 елементів.

Потім масив рекомендацій зберігається в файл model.p за допомогою бібіотеки pickle.

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

На головній сторінці показан хедер сайту, на якому присутні кнопки для переходу на сторінки реєстрації та авторизації в системі. Далі показано список жанрів. Якщо вибрати якийсь з них, то система відбере фільми тільки конкретного жанру. Далі, власне, ми бачимо сам список фільмів та пагінацію, за допомогою якої ми можемо завантажувати наступні фільми.

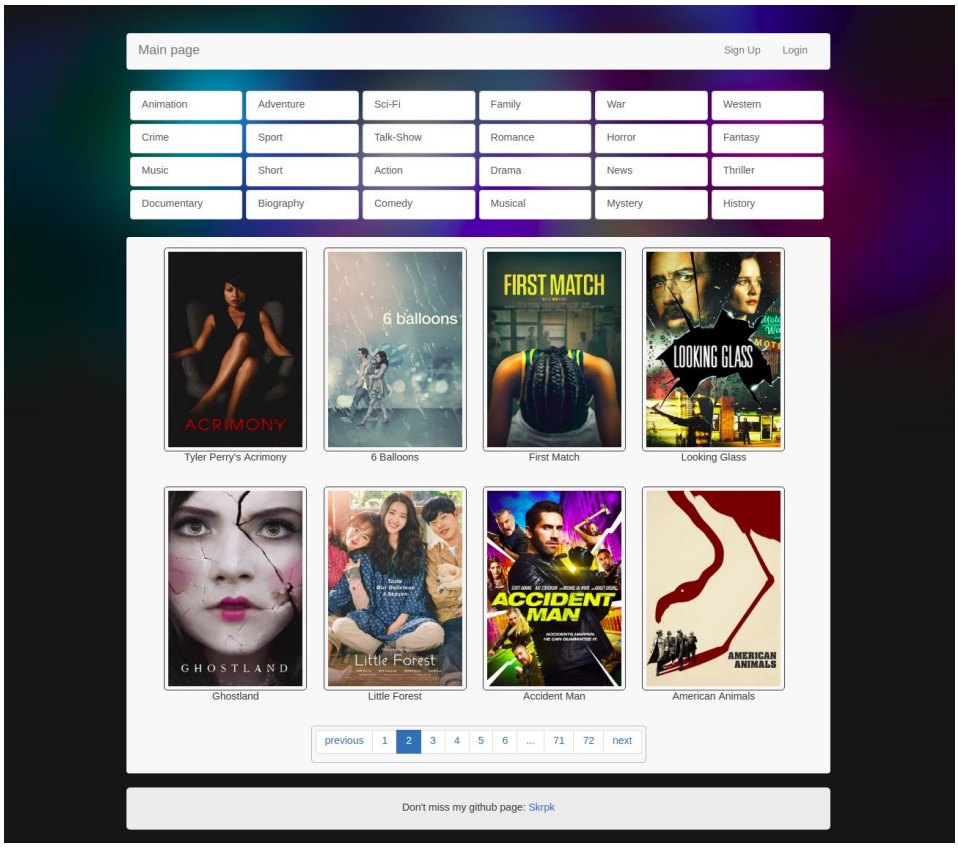

Рисунок 5.1 – Головна сторінка сайту

Якщо натиснути на на постер або назву певного фільму, то відкриється сторінка детального опису з датою виходу фільму, оцінками та описом. Також на сторінці можна переглянути коментарі від інших користувачів. Якщо відвідати цю сторінку, будучи авторизованим, то фільм також можна оцінити за 10-бальною шкалою або додати коментар.

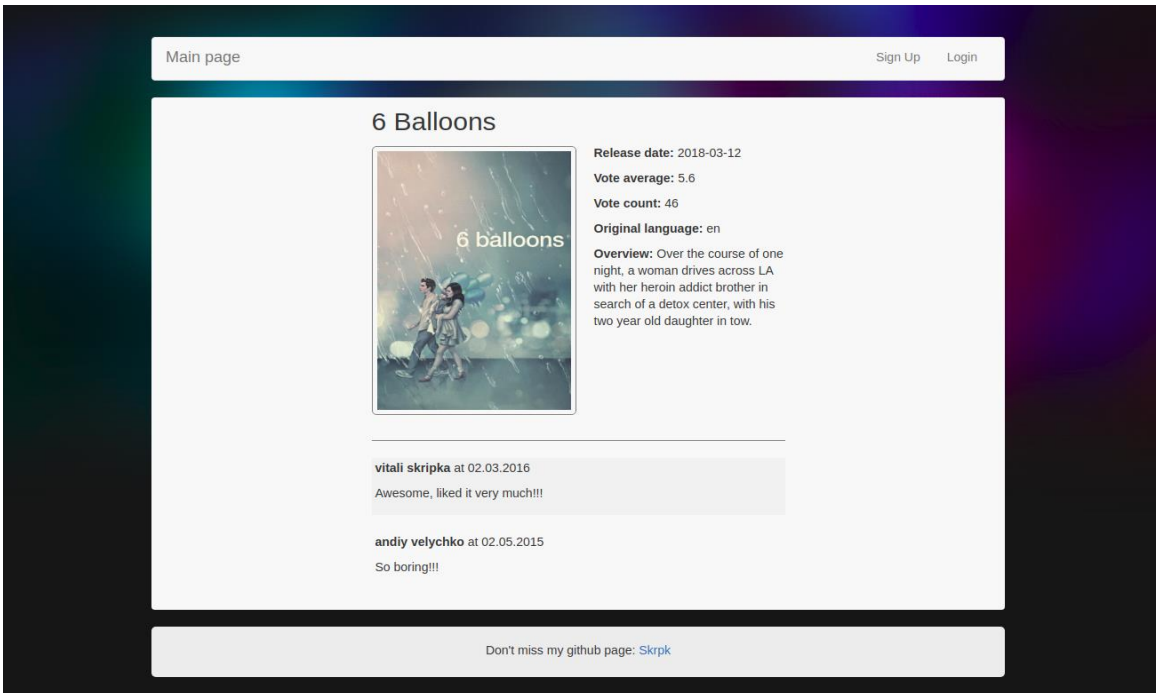

Рисунок 5.2 – Сторінка деталей фільму

При переході на сторінку авторизації з’явиться форма, в якій користувач вводить свою дані. Сторінка реєстрації не сильно відрізняється, від сторінки авторизації.

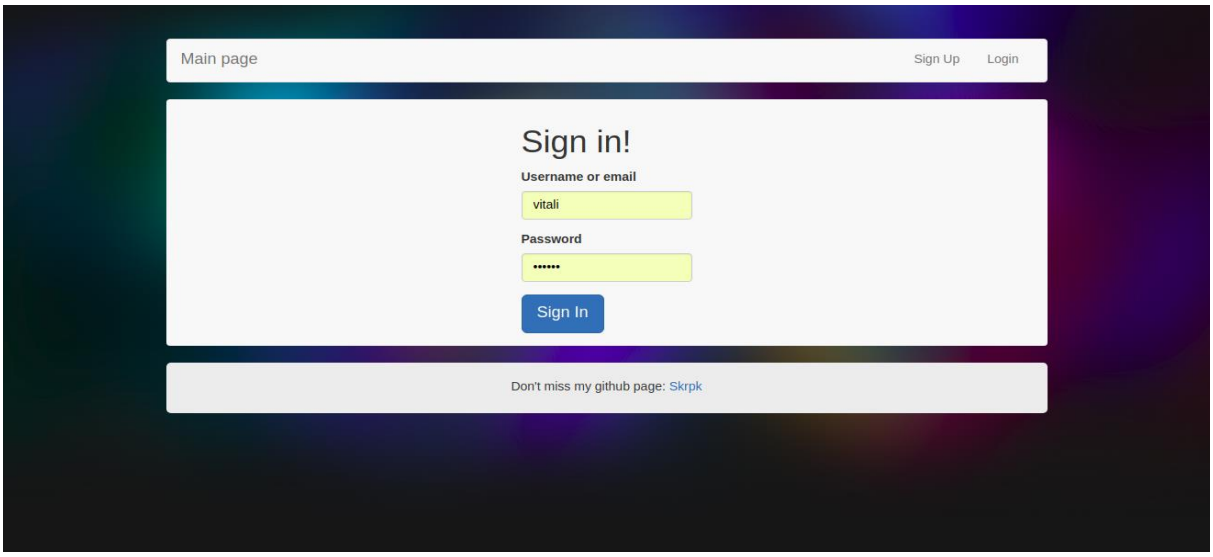

Рисунок 5.3 – Сторінка авторизації

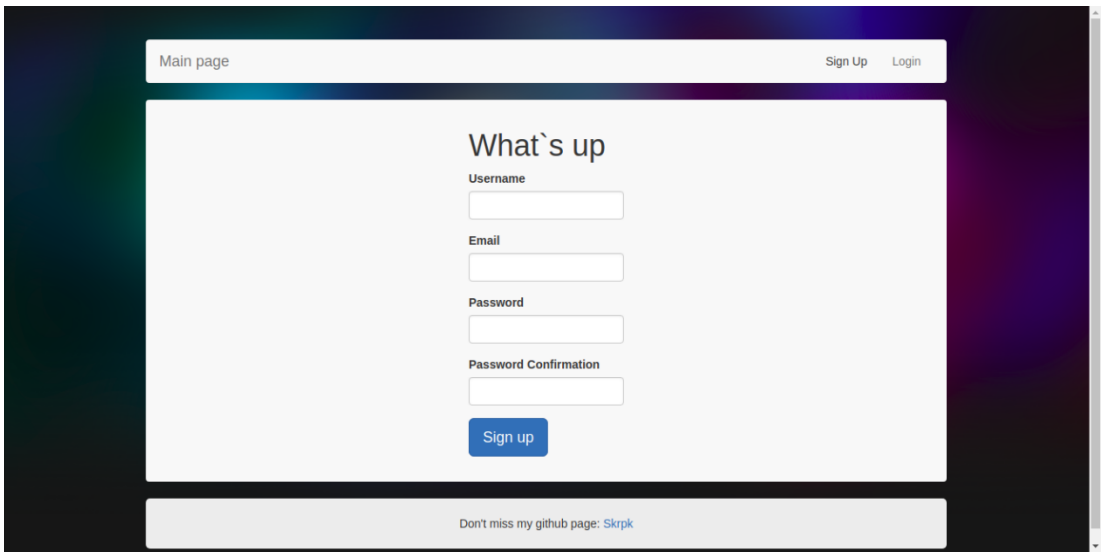

Рисунок 5.4 – Сторінка реєстрації

Авторизувавшись в системі, в хедері з’являться кнопки перегляду персональних рекомендацій та власного профілю.

На сторінці рекомендацій ми бачимо список фільмів, підібраних алгоритмом нейронної мережі.

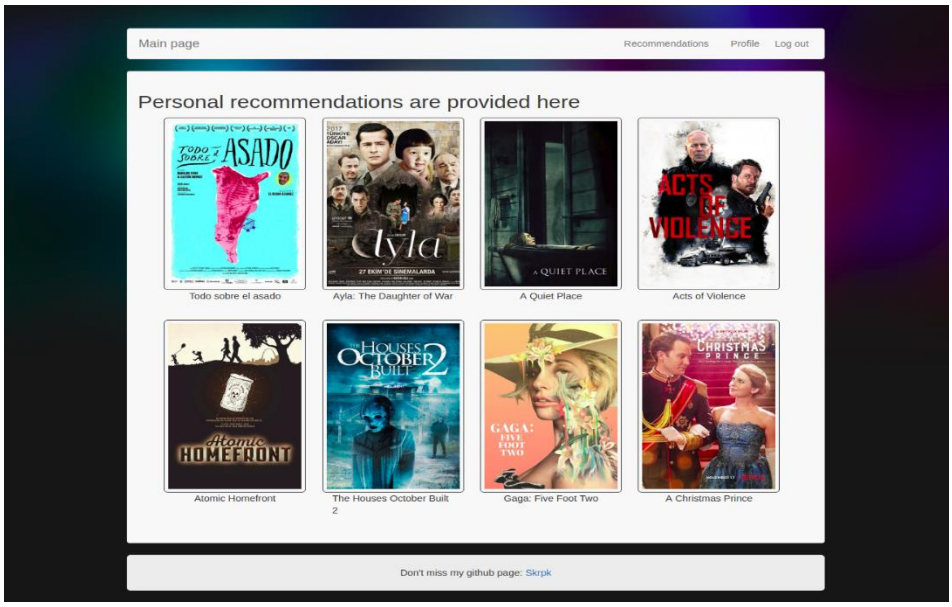

Рисунок 5.5 – Сторінка рекомендацій

На сторінці профілю ми можемо переглянути дані користувача або змінити пароль.

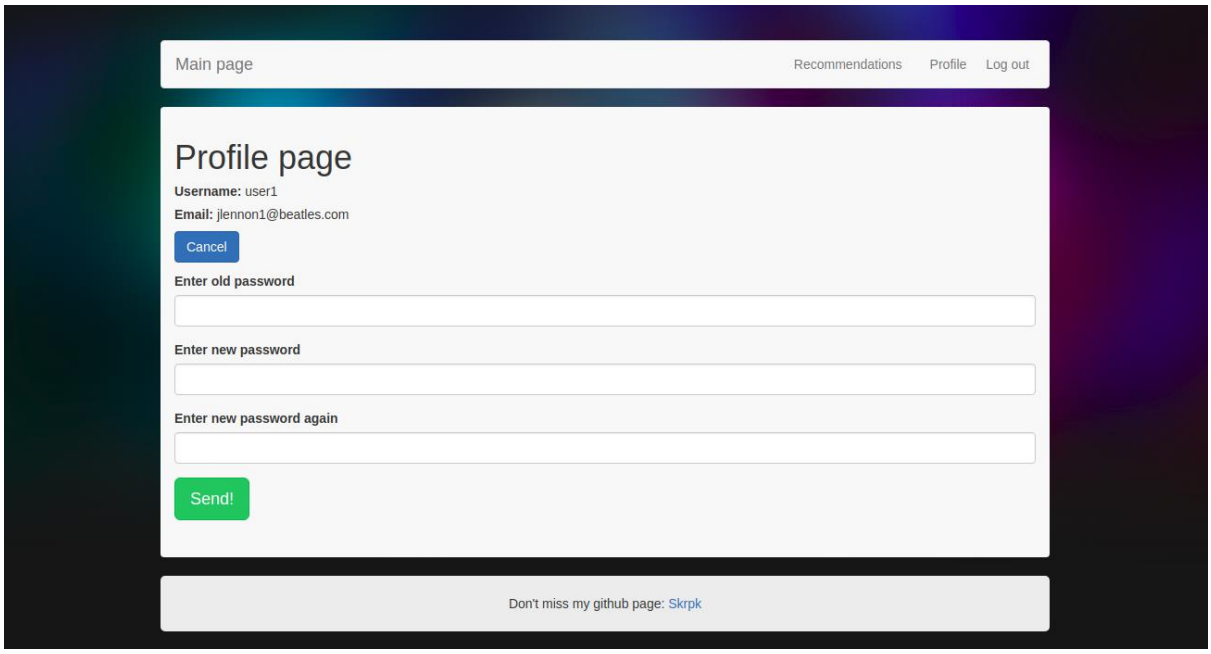

Рисунок 5.6 – Сторінка профілю

ВИСНОВКИ

У даній роботі були вивчені основні методи і алгоритми рекомендаційної системи. На їх основі реалізований прототип веб-сайту для побудови профілю та вибору кінофільму.

В ході даної роботи були вирішені наступні завдання:

- було досліджені способи і алгоритми рекомендаційних систем, їх недоліки та особливості;
- було досліджено особливості використання та імплементації нейронних мереж;
- здобуто навики використання React.js, Django, Python, PostgreSQL.
- спроектовано базу даних для збереження інформації про об’єкти, які використовуються для розрахунку рекомендацій;
- розроблено алгоритм рекомендацій;
- описані додаткові функції системи;
- розроблено прототип веб-сайту для пошуку та оцінювання фільмів;

В майбутньому планується розширити алгоритм рекомендацій за допомогою класифікації тексту коментарів нейронними мережами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Макогон Ю.О. Рекомендаційна система на базі нейронних мереж // Матеріали 25-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». – 2021. – 232 с.

2. Müller A. Introduction to Machine Learning with Python: A Guide for Data Scientists / A. Müller, S. Guido. – USA: O'Reilly Media, 2016. – 400 с.

3. Glassner A. Deep Learning: A Visual Approach / Andrew Glassner. – San Francisco: No Starch Press, 2021. – 776 с.

4. Lakshmanan V. Practical Machine Learning for Computer Vision: End-to-End Machine Learning for Images / V. Lakshmanan, M. Görner, R. Gillard. – Boston: O'Reilly Media, 2021. – 482 с.

5. PyCharm [Електронний ресурс] // jetbrains – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/.

6. What is Python? Executive Summary [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/doc/essays/blurb/.

7. What is NumPy? [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/stable/user/whatisnumpy.html.

8. Why TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/about.

9. About - OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/about/.

10. Aggarwal C. Neural Networks and Deep Learning: A Textbook / Charu C Aggarwal. – NY: Springer, 2018. – 520 с.

11. Mohit Sewak. Practical Convolutional Neural Networks: Implement advanced deep learning models using Python / S. Mohut, K. Md. Rezaul, P, Pradeep. – Birmingham: Packt Publishing, 2018. – 218 с.

12. Hands-On Convolutional Neural Networks with TensorFlow: Solve

computer vision problems with modeling in TensorFlow and Python / [I. Zafar, G. Tzanidou, R. Burton та ін.]. – Birmingham: Packt Publishing, 2018. – 274 с.

13. Kostadinov S. Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modeling with TensorFlow / S. Kostadinov. – Birmingham: Packt Publishing, 2018. – 122 с.

14. Babcock J. Generative AI with Python and TensorFlow 2: Create images, text, and music with VAEs, GANs, LSTMs, Transformer models / J. Babcock, R. Bali. – Birmingham: Packt Publishing, 2021. – 488 с.