Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Проєкт рекомендаційної системи фільмів на основі архітектури Netflix

Виконав(ла): студент(ка) 6 курсу, групи СНм-61
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Налутка П.В. (прізвище та ініціали)

Керівник (підпис) Палка О.В. (прізвище та ініціали)

Нормоконтроль (підпис) Дуда О.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"17" листопада 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Налутка Павло Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи **Проєкт рекомендаційної системи фільмів на основі архітектури Netflix.**

Керівник роботи **PhD., асист. Палка О.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "27" листопада 2025 року № 4/7-1042

2. Термін подання студентом завершеної роботи 21 грудня 2025 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

ВСТУП; 1 ОГЛЯД ПЛАТФОРМИ NETFLIX; 1.1 Короткий опис платформи Netflix та її рекомендаційної системи; 1.2 Основні принципи вироблення рекомендацій з точки зору клієнта платформи; 2 АНАЛІЗ ПРЕДМЕТНОЇ ОБАЛСТІ;2.1 Використання мовних моделей для рекомендаційної системи;2.2 Обговорення архітектури моделі;2.3 Виклики для пропонованої системи рекомендацій;2.4 Застосунки відтворення та проблеми їх використання; 2.5 Масштабування базових м оделей для рекомендацій Netflix; 3 ОПИС ПРОЄКТУ ПРОПОНОВАНОЇ СИСТЕМИ;3.1 Зв'язок рейтингу та рекомендацій. Ранжування;3.2 Дані взаємодії користувачів платформи та моделі побудови рекомендацій;3.3 Наука про дані від споживачів послуг;3.4 Дослідження даних та формування гіпотез; 3.5 Розробка та тестування системи;4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; 4.1 Питання щодо охорони праці; 4.2 Питання щодо безпеки в надзвичайних ситуаціях;ВИСНОВКИ;СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ;ДОДАТКИ

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Гурик О.Я, к.т.н., доц. каф. МТ		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 17 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	17.11.25-18.11.25	Виконано
2.	Підбір наукових джерел за темою роботи	19.11.25-20.11.25	Виконано
3.	Переклад та опрацювання наукових джерел	20.11.25-23.11.25	Виконано
	за темою кваліфікаційної роботи		
4.	Виконання дослідження щодо теми	24.11.25-10.12.25	Виконано
	кваліфікаційної роботи		
5.	Оформлення першого розділу	11.12.25-12.10.25	Виконано
6.	Оформлення другого розділу	12.12.25-13.12.25	Виконано
7.	Оформлення третього розділу	13.12.25-14.12.25	Виконано
8.	Виконання завдання до підрозділу "Охорона праці"	08.12.25-09.11.25	Виконано
9.	Виконання завдання до підрозділу "Безпека в	10.12.25-12.12.25	Виконано
	надзвичайних ситуаціях"		
10.	Оформлення кваліфікаційної роботи	12.12.25-31.12.25	Виконано
11.	Нормоконтроль	14.12.25-15.12.25	Виконано
12.	Перевірка на плагіат	15.12.25	Виконано
13.	Попередній захист кваліфікаційної роботи	18.12.25	Виконано
14.	Захист кваліфікаційної роботи	22.12.2025	

Студент _____ Налутка П.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Палка О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Проєкт рекомендаційної системи фільмів на основі архітектури Netflix" // Кваліфікаційна робота освітнього рівня "Магістр" // Налутка Павло Васильович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНм-61 // Тернопіль, 2025 // с. – 52, рис. – 16, табл. – 2, джерел – 22.

Ключові слова: рекомендаційна система, стрімінгова платформа, програмна архітектура, машинне навчання, персоналізація

Кожна стрімінгова платформа прагне допомогти користувачам знаходити фільми, які їм справді сподобаються. Для цього провайдери послуг створюють високорівневу систему рекомендацій (рекомендаційну систему). Її мета – прогнозувати, чи сподобається замовникові певний фільм, виходячи з того, як він оцінив інші кінострічки. На основі цих прогнозів провайдер формує індивідуальні добірки фільмів відповідно до вподобань кожного окремого користувача.

Попри те, що кожен провайдер реалізує алгоритм рекомендацій по своєму і вони працюють досить ефективно, у них все ж є потенціал для вдосконалення. Сьогодні існує багато перспективних методів, які могли б стати альтернативою поточним підходам, хоча частину з них компанії ще не випробовували. Наукові дослідження в цьому напрямку досить незначні, хоча дослідження власне алгоритмів рекомендацій є і серед дослідників ТНТУ і не тільки. Нас цікавить, чи здатен будь-який із рекомендаційних методів покращити використовувані алгоритми і забезпечити точніші прогнози. Для аналізу оберемо рекомендаційну систему на основі архітектури Netflix.

ANNOTATION

"A Project for a Movie Recommendation System Based on Netflix Architecture" // Master's degree qualification paper // Nalutka Pavlo // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНм-61 // Ternopil, 2025 // p. – 52, fig. – 16, tables – 2, references – 22.

Key words: recommendation system, streaming platform, software architecture, machine learning, personalization

Each streaming platform aims to help users find movies they will really like. To do this, service providers create a high-level recommendation system (recommendation system). Its goal is to predict whether a customer will like a particular movie, based on how he rated other films. Based on these predictions, the provider forms individual movie selections according to the preferences of each individual user.

Even though each provider implements the recommendation algorithm in its own way and they work quite effectively, they still have the potential for improvement. Today, there are many promising methods that could become an alternative to current approaches, although some of them have not yet been tested by companies. Scientific research in this direction is quite insignificant, although research into recommendation algorithms is also carried out among researchers at TNTU and beyond. We are interested in whether any of the recommendation methods can improve the algorithms used and provide more accurate forecasts. For analysis, we will choose a recommendation system based on the Netflix architecture.

ЗМІСТ

ВСТУП

Актуальність задачі.

Персоналізована система рекомендацій Netflix – це складна система, що вирізняється різноманітними спеціалізованими моделями машинного навчання, кожна з яких задовольняє різні потреби, включаючи "Продовжити перегляд" та "Найкращий вибір сьогодні для вас" [8]. Однак, оскільки ставиться за мету розширити набір алгоритмів персоналізації для задоволення зростаючих потреб бізнесу, обслуговування системи рекомендацій стане досить дорогим. Крім того, важко переносити інновації з однієї моделі в іншу, враховуючи, що більшість із них навчаються незалежно, незважаючи на використання спільних джерел даних. Цей сценарій підкреслив необхідність нової архітектури системи рекомендацій, де навчання вподобань учасників є централізованим, що підвищує доступність та корисність у різних моделях.

Зокрема, ці моделі переважно отримують ознаки з історії нещодавньої взаємодії учасників на платформі. Однак багато з них обмежені коротким часовим вікном через обмеження затримки обслуговування або вартості навчання. Це обмеження було основою для розробки базової моделі для рекомендацій. Ця модель має на меті асимілювати інформацію як з повної історії взаємодії учасників, так і з всього контенту у дуже великих масштабах. Вона сприяє поширенню цих знань на інші моделі, або через спільні вагові коефіцієнти моделі для точного налаштування, або безпосередньо через вбудовування [1, 3, 5].

Мета роботи.

Метою роботи було дослідити та оцінити альтернативні підходи до побудови системи рекомендацій фільмів з метою визначення, чи можуть вони забезпечити точніші прогнози користувацьких вподобань порівняно з існуючою системою, а також оцінити їхній потенційний вплив на користувацький досвід і бізнес-результати.

Для досягнення мети потрібно дослідити та дати відповіді на питання дослідження:

1. Які сучасні методи та алгоритми рекомендаційних систем, описані в науковій літературі або неформально запропоновані, можуть бути застосовані як альтернативи існуючим алгоритмам?
2. Чи забезпечують альтернативні підходи вищу точність прогнозування вподобань користувачів порівняно з існуючою системою?
3. Яким може бути вплив впровадження більш точної рекомендаційної моделі на користувацький досвід та бізнес-показники?
4. Передбачити рейтинг, який користувач дав би фільму, який він ще не оцінив.
5. Мінімізувати різницю між прогнозованим і фактичним рейтингом (RMSE та MAPE).

Об’єкт дослідження: процеси розробки алгоритмів рекомендаційної системи стрімінгової платформи.

Предмет дослідження: платформи системи рекомендацій фільмів від компанії Netflix.

Наукова новизна отриманих результатів.

Проведено емпіричне дослідження літературних джерел, щоб знайти фактори, які мають позитивний або негативний вплив на якість роботи рекомендаційної системи фільмів Netflix. Отримані результати дають корисні рекомендації щодо розробки відповідних алгоритмів.

Практичне значення отриманих результатів.

У цій роботі розглянуті питання впровадження алгоритмів реалізації рекомендацій фільмів на основі існуючої системи Netflix. На концептуальному рівні запропоновано основні архітектурні рішення.

Апробація результатів та особистий внесок здобувача.

Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету імені Івана Пулюя. Результати кваліфікаційної роботи опубліковані

у тезах студентської наукової конференції "Актуальні задачі сучасних технологій – 2025", яка проводилась у ТНТУ.

1 ОГЛЯД ПЛАТФОРМИ NETFLIX

1.1 Короткий опис платформи Netflix та її рекомендаційної системи

Одна з причин, чому фокус в алгоритмах рекомендацій повинен змінитися, полягає в тому, що Netflix в цілому кардинально змінився за останні кілька років. Netflix запустив сервіс миттєвого потокового передавання у 2007 році. Потокове передавання змінило не лише спосіб взаємодії учасників із сервісом, але й тип даних, доступних для використання в алгоритмах. Спочатку сервіс розповсюджував фільми на дискових носіях – DVD-дисках. Щодо DVD, метою було допомогти людям заповнити свою чергу фільмами, які вони отримають поштою протягом наступних днів і тижнів; вибір відбувається віддалено в часі від перегляду, люди ретельно вибирають, оскільки обмін DVD на інший займає більше дня, і компанія не отримує зворотного зв’язку під час перегляду. Щодо потокового передавання, учасники шукають щось чудове для перегляду прямо зараз; вони можуть переглянути кілька відео, перш ніж зупинитися на одному, вони можуть переглянути кілька за один сеанс, і тоді можна спостерігати за статистикою переглядів, наприклад, чи було відео переглянуто повністю чи лише частково.

Ще однією великою зміною став перехід від одного веб-сайту до сотень пристроїв. Інтеграцію з плеєром Roku та Xbox було оголошено у 2008 році, через два роки після початку конкуренції з Netflix. Всього через рік потокове передавання Netflix потрапило на iPhone. Тепер воно доступне на безлічі пристроїв, від Android до найновішого AppleTV.

Сьогодні Netflix має понад 23 мільйони передплатників у 47 країнах. Ці передплатники переглядали 2 мільярди годин трансляцій із сотень різних пристроїв в останньому кварталі 2011 року. Щодня вони додають до черги 2 мільйони фільмів і телешоу та генерують 4 мільйони рейтингів.

Компанія адаптувала свої алгоритми персоналізації до цього нового сценарію таким чином, що тепер 75% того, що люди дивляться, походить з

певних рекомендацій. Цього було досягнуто завдяки постійній оптимізації досвіду учасників і зафіксовано значне зростання задоволеності учасників щоразу, коли персоналізація покращувалася для замовників. Опишемо основні методи та підходи, які використовуються для створення цих рекомендацій.

1.2 Основні принципи вироблення рекомендацій з точки зору клієнта платформи

Протягом багатьох років роботи системи виявилося, що для підписників сервісу надзвичайно цінно включати рекомендації для персоналізації якомога більшої частини пропонованого контенту. Персоналізація починається на нашій головній сторінці, яка складається з груп відео, розташованих горизонтальними рядами. Кожен рядок має заголовок, який передає задуманий змістовний зв’язок між відео в цій групі. Більшість персоналізації базується на тому, як здійснюється вибір стрічок, як ми визначаємо, які елементи в них включати, і в якому порядку розміщувати ці елементи.

Проаналізуємо для початку стрічку рекомендацій "Топ-10". Це найкраща для сервісу оцінка десяти назв, які найбільше сподобаються клієнтові. Важливо пам’ятати, що персоналізація Netflix призначена для роботи з родиною, в якій, ймовірно, проживають різні люди з різними смаками. Саме тому, коли клієнт бачить свій "Топ-10", то, ймовірно, там знайдуться товари для батька, матері, дітей або всієї родини. Щоб досягти цього, у багатьох частинах системи Netflix виконується оптимізація не лише точності, але й різноманітності (див. рис. 1.1).

Ще одним важливим елементом персоналізації Netflix є обізнаність. Компанія прагне, щоби учасники знали, як вона адаптується до їхніх смаків. Це не лише сприяє довірі до системи, але й заохочує учасників надавати відгуки, які призведуть до кращих рекомендацій. Інший спосіб підвищення довіри за допомогою компонента персоналізації – це надання пояснень щодо того, чому ми вирішили рекомендувати певний фільм чи шоу. Рекомендується це не тому, що він відповідає бізнес-потребам компанії, а тому, що він відповідає інформації,

яку ми маємо від клієнтів: їх явним смаковим уподобанням та оцінкам, історії переглядів або навіть рекомендаціям друзів.

Рисунок 1.1 – Приклад стрічки рекомендацій "Топ-10"

Щодо друзів, нещодавно платформа запустила функцію зв’язку з Facebook, де працює платформа. Відомості про друзів клієнта в соціальних мережах не лише дає ще один сигнал для використання в алгоритмах персоналізації, але й дозволяє використовувати різні рядки, які здебільшого покладаються на коло спілкування замовника, для створення рекомендацій (див. рис. 1.2).

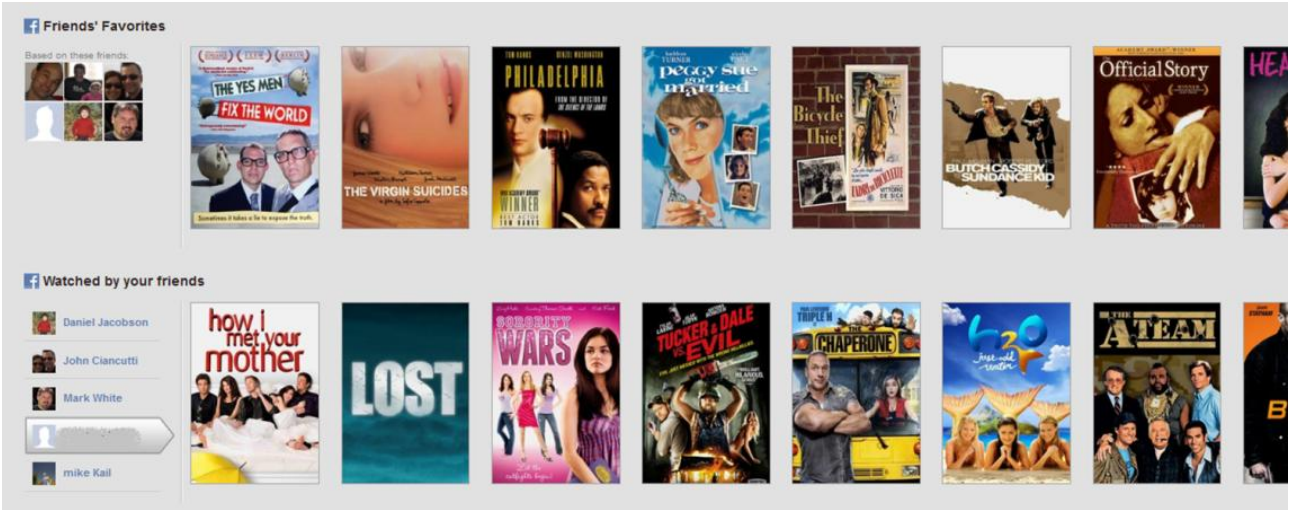


Рисунок 1.2 –Зв'язок із соціальною мережею Facebook

Однією з найбільш впізнаваних персоналізацій у нашому сервісі є колекція рядків "жанрів". Вони варіюються від знайомих високорівневих категорій, таких як "Комедії" та "Драми", до високоспеціалізованих зрізів, таких як "Фільми про подорожі в часі з 1980-х років". Кожен рядок представляє 3 рівні персоналізації: вибір самого жанру, підмножину назв, вибраних у цьому жанрі, та рейтинг цих назв. Учасники настільки добре взаємодіють із цими рядками, що алгоритми рекомендацій вимірюють збільшення утримання учасників, розміщуючи найбільш персоналізовані рядки вище на сторінці, а не нижче. Як і з іншими елементами персоналізації, свіжість та різноманітність враховуються при виборі жанрів, які показувати з тисяч можливих (див. рис. 1.3).

Рисунок 1.3 – Уточнення персональних рекомендацій за жанрами

Пояснення вибору рядків здійснюється, використовуючи неявні жанрові вподобання учасника – нещодавні п'єси, рейтинги та інші взаємодії – або явні

відгуки, надані через наше опитування щодо смакових уподобань. Також учасникам пропонується зосередити увагу на рядку з додатковими явними відгуками про вподобання, якщо їх немає (див. рис. 1.4).

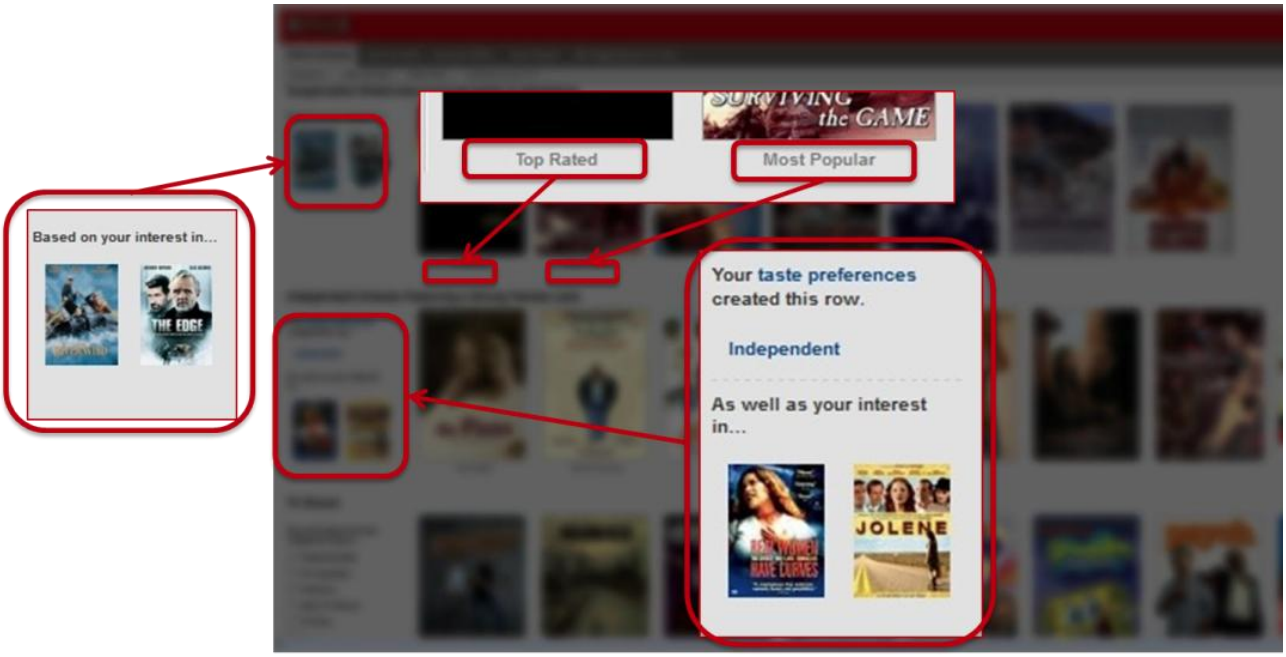


Рисунок 1.4 – Персональна стрічка рекомендацій
на основі відгуків клієнта

Схожість також є важливим джерелом персоналізації в сервісі. Платформа розглядає подібність у дуже широкому сенсі; вона може бути між фільмами або між учасниками, і може мати кілька вимірів, таких як метадані, рейтинги або дані переглядів. Крім того, ці подібності можна поєднувати та використовувати як ознаки в інших моделях. Схожість використовується в різних контекстах, наприклад, у відповідь на дії учасника, такі як пошук або додавання фільму до черги. Вона також використовується для створення рядків "спеціальних жанрів" на основі подібності до фільмів, з якими учасник нещодавно взаємодіяв.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБАЛСТІ

2.1 Використання мовних моделей для рекомендаційної системи

Поштовхом для побудови базової моделі рекомендацій є зсув парадигми в обробці природної мови (NLP) до моделей великих мов (LLM). У NLP спостерігається тенденція відходу від численних малих, спеціалізованих моделей до єдиної моделі великої мовної мови, яка може виконувати різноманітні завдання безпосередньо або з мінімальним налаштуванням. Ключові висновки з цього зсуву включають:

1. Дата-центричний підхід: зміщення фокусу з модельно-центричних стратегій, які значною мірою залежать від інженерії ознак, на дата-центричний. Цей підхід надає пріоритет накопиченню великомасштабних, високоякісних даних і, де це можливо, спрямований на наскрізне навчання.

2. Використання напівавторизованого навчання – прогнозування наступного токена в LLM виявилося надзвичайно ефективним. Воно дозволяє проводити масштабне напівавторизоване навчання з використанням немаркованих даних, а також забезпечує модель напрочуд глибоким розумінням знань.

Ці висновки вплинули на дизайн нашої базової моделі, що дозволило перейти від підтримки численних невеликих спеціалізованих моделей до побудови масштабованої, ефективної системи. Масштабуючи напівавторизовані навчальні дані та параметри моделі, ми прагнемо розробити модель, яка не лише відповідає поточним потребам, але й динамічно адаптується до мінливих вимог, забезпечуючи стійкі інновації та ефективне використання ресурсів.

У Netflix залучення користувачів охоплює широкий спектр, від звичайного перегляду веб-сторінок до цілеспрямованого перегляду фільмів. З понад 300 мільйонами користувачів на кінець 2024 року це перетворюється на сотні мільярдів взаємодій – величезний набір даних, порівнянний за масштабом з обсягом токенів моделей великих мов програмування (LLM). Однак, як і в LLM,

якість даних часто переважає їх самий обсяг. Щоб ефективно використовувати ці дані, ми застосовуємо процес токенізації взаємодій, що забезпечує виявлення значущих подій та мінімізацію надлишків.

Токенізація взаємодій з користувачами. Не всі необроблені дії користувача однаково сприяють розумінню уподобань. Токенізація допомагає визначити, що являє собою значущий "токен" у послідовності. Проводячи аналогію з байтовим парним кодуванням (BPE) в NLP, ми можемо розглядати токенізацію як об'єднання суміжних дій для формування нових токенів вищого рівня. Однак, на відміну від токенізації мови, створення цих нових токенів вимагає ретельного розгляду того, яку інформацію зберігати. Наприклад, може знадобитися підсумувати загальну тривалість перегляду або агрегувати типи взаємодії, щоб зберегти критичні деталі (див. рис. 2.1).

Рисунок 2.1 – Токенізація історії взаємодії з користувачем шляхом об'єднання дій над одним і тим самим заголовком

Цей компроміс між гранулярними даними та стисненням послідовностей подібний до балансу в LLM між розміром словника та контекстним вікном. У нашому випадку метою є балансування тривалості історії взаємодії з рівнем деталізації, що зберігається в окремих токенах. Надмірно узагальнена токенізація ризикує втратою цінних сигналів, тоді як занадто гранулярна послідовність може перевищити практичні обмеження часу обробки та пам'яті.

Навіть за таких стратегій історії взаємодії активних користувачів можуть охоплювати тисячі подій, перевищуючи можливості моделей-трансформерів зі стандартними шарами самоуваги.

Самоувага – це тип механізму уваги, що використовується в моделях машинного навчання. Цей механізм використовується для оцінки важливості токенів або слів у вхідній послідовності для кращого розуміння зв'язків між ними. Це ключова частина трансформаторних моделей, потужної архітектури штучного інтелекту, яка необхідна для завдань обробки природної мови (NLP). Архітектура трансформатора є основою для більшості сучасних моделей великих мов (LLM).

Механізм самоуваги був введений за допомогою трансформатора, архітектури модельної нейронної мережі, запропонованої дослідниками [9]. Метою запропонованої архітектури було вирішення проблем традиційних моделей машинного навчання, які використовують згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN). У рекомендаційних системах контекстні вікна під час виведення часто обмежені сотнями подій – не через можливості моделі, а тому, що ці сервіси зазвичай вимагають затримки на рівні мілісекунд. Це обмеження є більш суворим, ніж типово для застосувань LLM, де довший час виведення (секунди) є більш допустимим.

Щоб вирішити цю проблему під час навчання, ми впроваджуємо два ключові рішення:

1. Механізми розрідженої уваги. Використовуючи методи розрідженої уваги, такі як низькорангове стиснення, модель може розширити своє контекстне вікно до кількох сотень подій, зберігаючи при цьому обчислювальну ефективність. Це дозволяє їй обробляти ширші історії взаємодії та отримувати глибше розуміння довгострокових уподобань.

2. Вибірка ковзного вікна [10]: під час навчання ми вибираємо перекриваючі вікна взаємодій з повної послідовності. Це гарантує, що модель піддається впливу різних сегментів історії користувача протягом кількох епох,

дозволяючи їй навчатися з усієї послідовності без необхідності використання непрактично великого контекстного вікна.

Під час виведення даних, коли потрібне багатоетапне декодування, ми можемо розгорнути кешування KV для ефективного повторного використання минулих обчислень та підтримки низької затримки.

KV кешування (Key-Value Caching) – це метод у трансформерних моделях, який дозволяє зберігати проміжні обчислення у вигляді векторів ключів (K) та значень (V) для вже згенерованих токенів [11]. Такий підхід усуває необхідність повторного перерахунку всієї послідовності на кожному кроці генерації, що суттєво прискорює інференс і переводить його з квадратичної складності до лінійної. Модель фактично "пам’ятає" попередній контекст, використовуючи закешовані K та V, і виконує обчислення уваги лише для нового токена.

Робота механізму виглядає так. На першому кроці модель формує K та V для початкового токена і зберігає їх у кеші. Далі, при генерації нового токена, він виступає як запит (Query), а модель використовує вже збережені ключі та значення для всіх попередніх токенів. Обчислюється увага, після чого нові K та V додаються до кешу. Таким чином, замість повторного аналізу всієї послідовності, система працює лише з новими даними.

Основні переваги – значне прискорення інференсу, особливо при довгих послідовностях, та ефективність у використанні обчислювальних ресурсів. Недоліки полягають у тому, що кеш зростає пропорційно довжині контексту, споживаючи більше пам’яті (зокрема VRAM), а також потребує оптимального керування, іноді із застосуванням технік вивантаження на CPU.

Ці підходи разом дозволяють нам збалансувати потребу в детальному, довгостроковому моделюванні взаємодії з практичними обмеженнями навчання моделей та логічного висновку, підвищуючи як точність, так і масштабованість нашої системи рекомендацій.

Інформація в кожному "токені". Хоча перша частина нашого процесу токенізації зосереджена на структуруванні послідовностей взаємодій, наступним критичним кроком є визначення цінної інформації, що міститься в кожному

токені. На відміну від LLM, які зазвичай покладаються на один простір вбудовування для представлення вхідних токенів, наші події взаємодії містять багато різнорідних деталей. До них належать атрибути самої дії (такі як локалізація, час, тривалість і тип пристрою), а також інформація про контент (така як ідентифікатор елемента та метадані, такі як жанр і країна випуску). Більшість цих функцій, особливо категоріальні, безпосередньо вбудовані в модель, що використовує наскрізний підхід до навчання. Однак деякі функції потребують особливої уваги. Наприклад, часові позначки потребують додаткової обробки для фіксації як абсолютних, так і відносних понять часу, причому абсолютний час особливо важливий для розуміння поведінки, залежної від часу.

Щоб підвищити точність прогнозування в послідовних системах рекомендацій, ми об'єднуємо характеристики токенів у дві категорії:

1. Функції часу запиту – це функції, доступні на момент прогнозування, такі як час входу, пристрій або місцезнаходження.

2. Функції після дії – це деталі, доступні після того, як відбулася взаємодія, такі як конкретний серіал, з яким здійснилася взаємодія, або тривалість взаємодії.

Щоб передбачити наступну взаємодію, ми поєднуємо ознаки часу запиту з поточного кроку з ознаками після дії з попереднього кроку [12]. Таке поєднання контекстуальної та історичної інформації гарантує, що кожен токен у послідовності має повне представлення, що фіксує як безпосередній контекст, так і моделі поведінки користувача з плином часу.

2.2 Обговорення архітектури моделі

Як згадувалося раніше, наш підхід за замовчуванням використовує ціль авторегресивного прогнозування наступного токена, подібну до GPT. Ця стратегія ефективно використовує величезний масштаб немаркованих даних про взаємодію з користувачем. Впровадження цієї цілі в системах рекомендацій продемонструвало численні успіхи [13–15]. Однак, враховуючи чіткі відмінності

між мовними завданнями та завданнями рекомендацій, введено кілька критичних змін до цілі.

По-перше, під час фази переднавчання типових LLM, таких як GPT, кожен цільовий токен зазвичай обробляється з однаковою вагою. Натомість, у нашій моделі не всі взаємодії з користувачами мають однакову важливість. Наприклад, 5-хвилинне відтворення трейлера не повинно мати таку ж вагу, як 2-годинний перегляд повного фільму. Більша проблема виникає при спробі узгодити довгострокову задоволеність користувачів з певними взаємодіями та рекомендаціями. Щоб вирішити цю проблему, ми можемо прийняти мету прогнозування для кількох токенів під час навчання, де модель передбачає наступні n токенів на кожному кроці замість одного токена [16]. Такий підхід заохочує модель фіксувати довгострокові залежності та уникати недалекоглядних прогнозів, зосереджених виключно на найближчих наступних подіях.

По-друге, ми можемо використовувати кілька полів у наших вхідних даних як допоміжні цілі прогнозування на додаток до прогнозування наступного ідентифікатора елемента, який залишається основною цільовою групою. Наприклад, ми можемо вивести жанри з елементів у вихідній послідовності та використовувати цю послідовність жанрів як допоміжну цільову групу. Цей підхід служить кільком цілям: він діє як регуляризатор для зменшення перенавчання при прогнозуванні ідентифікаторів елементів із зашумленістю, надає додаткове розуміння намірів користувача або довгострокових жанрових уподобань, а також, будучи структурованим ієрархічно, може покращити точність прогнозування ідентифікатора цільового елемента. Спочатку прогнозуючи допоміжні цілі, такі як жанр або оригінальна мова, модель ефективно звужує список кандидатів, спрощуючи наступне прогнозування ідентифікатора елемента.

2.3 Виклики для пропонованої системи рекомендацій

Окрім інфраструктурних проблем, що виникають під час навчання більших моделей зі значними обсягами даних про взаємодію з користувачем, що є поширеним явищем під час спроб побудови базових моделей, існує кілька унікальних перешкод, характерних для рекомендацій, які роблять їх життєздатними. Однією з унікальних проблем є холодний запуск сутності.

Нові фільми часто додаються до каталогу. Тому моделі базової рекомендації вимагають можливості холодного запуску, що означає, що моделі повинні оцінювати вподобання учасників щодо нещодавно запущених фільмів, перш ніж хтось взаємодіятиме з ними. Щоб забезпечити це, система навчання базової моделі побудована з такими двома можливостями: поетапне навчання та можливість робити висновки з невидимими об'єктами.

1. Поетапне навчання. Базові моделі навчаються на великих наборах даних, включаючи історію дій кожного учасника, що робить часте перенавчання непрактичним. Однак наш каталог та налаштування учасників постійно розвиваються. На відміну від моделей великих мов, які можна поетапно навчати за допомогою стабільних словників токенів, наші рекомендаційні моделі вимагають нових вбудовувань для нових назв, що вимагає розширених шарів вбудовування та вихідних компонентів. Щоб вирішити цю проблему, ми запускаємо нові моделі з "теплого" режиму, повторно використовуючи параметри з попередніх моделей та ініціалізуючи нові параметри для нових назв. Наприклад, нові вбудовування назв можна ініціалізувати, додаючи невеликий випадковий шум до існуючих усереднених вбудовувань або використовуючи зважену комбінацію вбудовувань подібних назв на основі метаданих. Такий підхід дозволяє новим назвам починати з відповідних вбудовувань, що сприяє швидшому точному налаштуванню. На практиці метод ініціалізації стає менш критичним, коли для точного налаштування використовується більше даних про взаємодію учасників.

2. Робота з невидимими сутностями. Навіть за умови поступового навчання не завжди гарантовано ефективне навчання на нових сутностях (наприклад, нещодавно запущених іграх). Також можливо, що деякі нові сутності не будуть включені/помічені в навчальних даних, навіть якщо ми часто налаштовуватимемо базові моделі. Тому важливо дозволити базовим моделям використовувати метадані сутностей та вхідних даних, а не лише дані про взаємодію учасників. Таким чином, наша базова модель поєднує як вбудовування ідентифікаторів елементів, що вивчаються, так і вбудовування, що вивчаються, з метаданих. Наступна діаграма на рисунку 2.2 демонструє цю ідею.

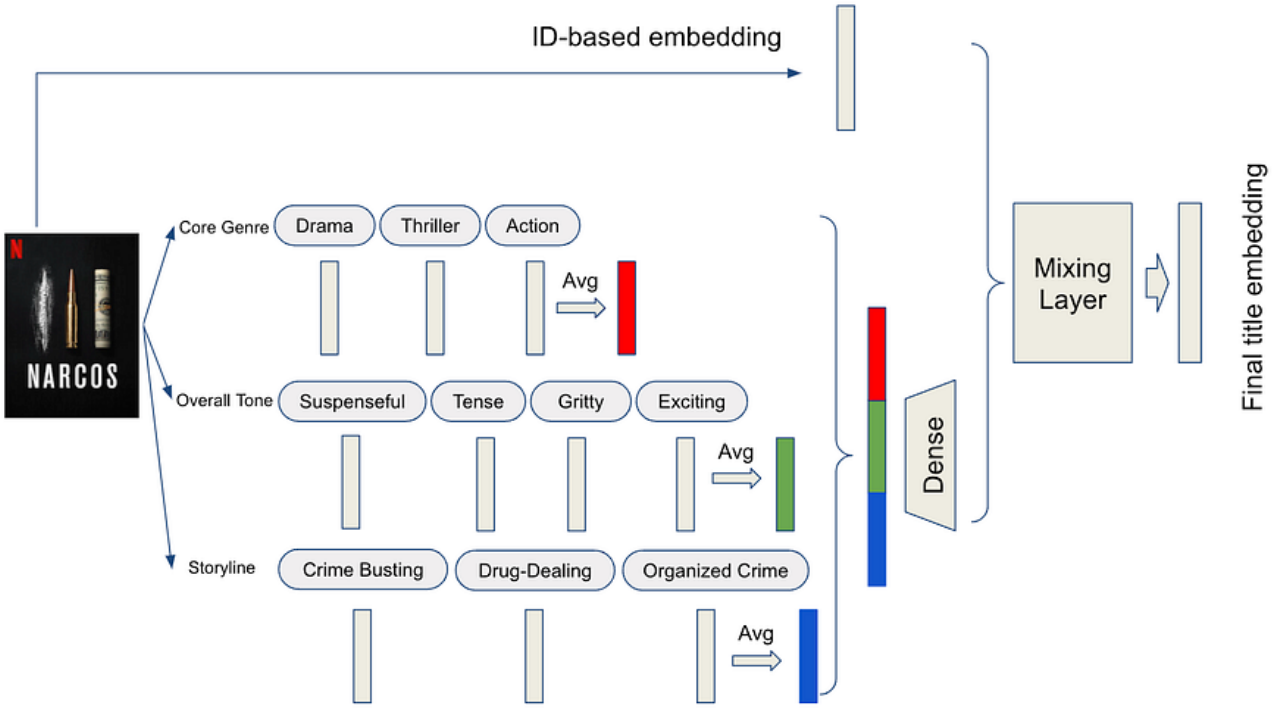


Рисунок 2.2 – Заголовки пов'язані з різними метаданими, такими як жанри, сюжетні лінії та тони

Кожен тип метаданих може бути представлений шляхом усереднення відповідних його вбудовувань, які потім об'єднуються для формування загального вбудовування на основі метаданих для заголовка. Щоб створити остаточне вбудовування заголовків, ми поєднуємо це вбудовування на основі метаданих із повністю навчальним вбудовуванням на основі ідентифікаторів за

допомогою шару змішування. Замість простого підсумовування цих вбудовувань ми використовуємо механізм уваги, заснований на "віку" сутності. Такий підхід дозволяє новим заголовкам з обмеженими даними взаємодії більше покладатися на метадані, тоді як усталені заголовки можуть більше покладатися на вбудовування на основі ідентифікаторів. Оскільки заголовки з подібними метаданими можуть мати різну залученість користувачів, їхні вбудовування повинні відображати ці відмінності. Введення певної випадковості під час навчання заохочує модель навчатися на метаданих, а не покладатися виключно на вбудовування ідентифікаторів. Цей метод гарантує, що нещодавно запущені або перед запуском заголовки матимуть прийнятну кількість вбудовувань навіть без даних про взаємодію з користувачем.

2.4 Застосунки відтворення та проблеми їх використання

Наша модель рекомендаційної бази розроблена для розуміння довгострокових уподобань учасників і може бути використана різними способами наступними додатками.

1. Безпосереднє використання як прогностичної моделі. Модель в першу чергу навчена прогнозувати наступний об'єкт, з яким взаємодіятиме користувач. Вона включає кілька головок прогностичних елементів для різних завдань, таких як прогнозування вподобань учасників щодо різних жанрів. Їх можна безпосередньо застосовувати для задоволення різноманітних бізнес- потреб.

2. Використання вбудовування. Модель генерує цінні вбудовування для членів та сутностей, таких як відео, ігри та жанри. Ці вбудовування обчислюються в пакетних завданнях та зберігаються для використання як в офлайн-, так і в онлайн-додатках. Вони можуть служити функціями в інших моделях або використовуватися для генерації кандидатів, таких як отримання привабливих назв для користувача. Високоякісні вбудовування назв також підтримують рекомендації від назви до назви. Однак одним важливим міркуванням є те, що простір вбудовування має довільні, неінтерпретовані

розміри та несумісний між різними навчальними запусками моделі. Це створює проблеми для подальших споживачів, які повинні адаптуватися до кожного перенавчання та повторного розгортання, ризикуючи помилками через недійсні припущення щодо структури вбудовування. Щоб вирішити цю проблему, ми застосовуємо ортогональне перетворення низького рангу для стабілізації простору вбудовування користувача/елемента, забезпечуючи узгоджене значення розмірів вбудовування, навіть коли базова модель перенавчається та повторно розгортається.

3. Точне налаштування за допомогою специфічних даних. Адаптивність моделі дозволяє точно налаштовувати її за допомогою специфічних для програми даних. Користувачі можуть інтегрувати повну модель або підграфи у власні моделі, точно налаштовуючи їх з меншими обсягами даних та обчислювальної потужності. Такий підхід досягає продуктивності, порівнянної з попередніми моделями, незважаючи на те, що початкова базова модель вимагає значних ресурсів.

2.5 Масштабування базових моделей для рекомендацій Netflix

Масштабуючи нашу базову модель для рекомендацій Netflix, ми базуємося на широкому та успішному застосуванні моделей великих мов (LLM). Так само, як LLM продемонстрували силу масштабування у покращенні продуктивності, виявляється, що масштабування має вирішальне значення для покращення завдань генеративних рекомендацій. Успішне масштабування вимагає надійної оцінки, ефективних алгоритмів навчання та значних обчислювальних ресурсів. Оцінка повинна ефективно диференціювати продуктивність моделі та визначати області для покращення. Масштабування включає масштабування даних, моделі та контексту, включаючи залучення користувачів, зовнішні огляди, мультимедійні ресурси та високоякісні вбудовування. Експерименти підтверджують, що закон масштабування також застосовується до нашої базової

моделі, причому спостерігаються постійні покращення зі збільшенням розміру даних та моделі (див. рис. 2.3).

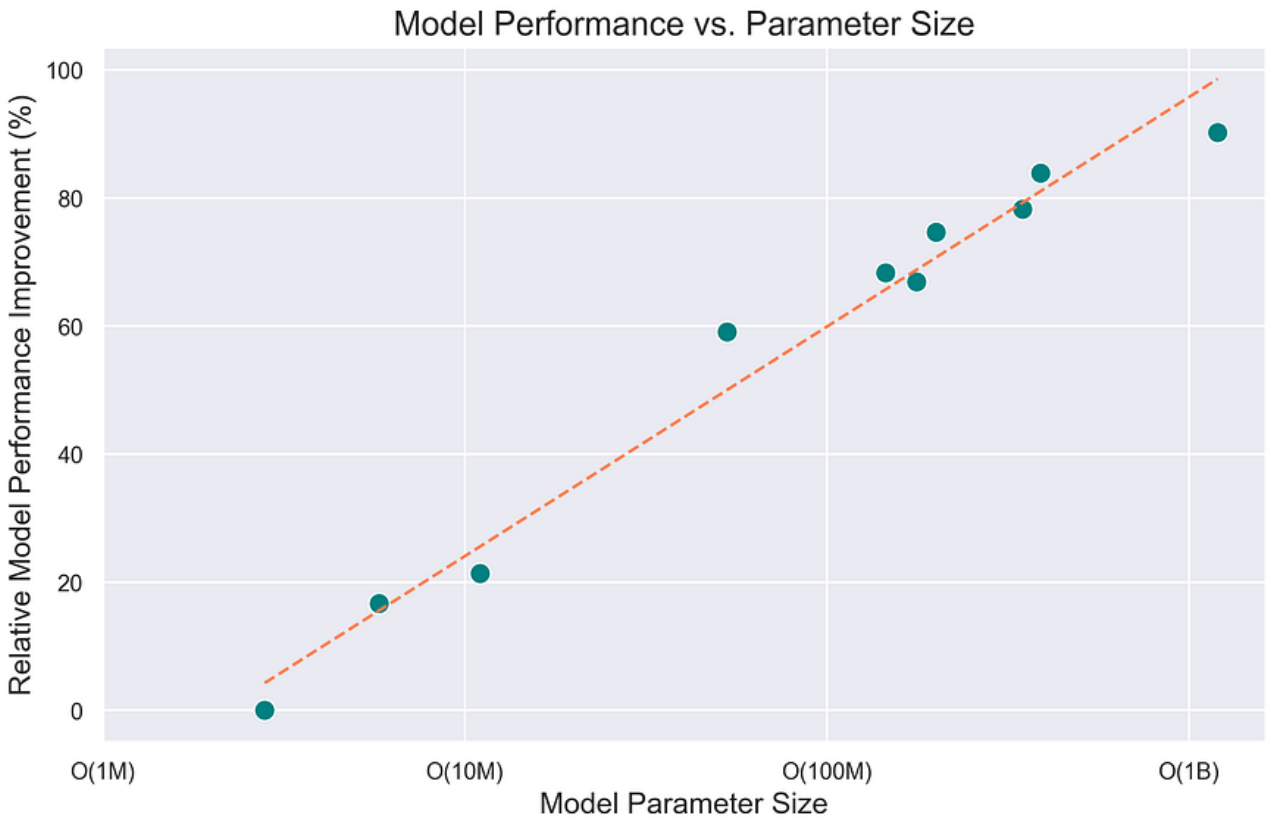


Рисунок 2.3 – Зв'язок між розміром параметра моделі та відносним покращенням продуктивності

Графік демонструє закон масштабування в моделюванні рекомендацій, показуючи тенденцію до підвищення продуктивності зі збільшенням розміру моделі. Вісь x має логарифмічне масштабування, щоб виділити зростання за різними величинами.

Таким чином базова модель для персоналізованих рекомендацій є значним кроком до створення єдиної, орієнтованої на дані системи, яка використовує великомасштабні дані для підвищення якості рекомендацій для наших учасників. Цей підхід запозичує знання з моделей великих мов (LLM), зокрема принципи напівавторизованого навчання та наскрізного навчання, прагнучи використати величезний масштаб немаркованих даних про взаємодію з користувачами. Вирішуючи унікальні проблеми, такі як холодний старт та упередженість

презентації, модель також визнає чіткі відмінності між мовними завданнями та рекомендаціями. Базова модель дозволяє використовувати різні наступні програми, від безпосереднього використання як прогностичної моделі до створення вбудовування користувачів та сутностей для інших програм, і може бути точно налаштована для конкретних полотен. Ми бачимо багатообіцяючі результати від наступних інтеграцій. Цей перехід від кількох спеціалізованих моделей до більш комплексної системи знаменує собою захопливий розвиток у галузі персоналізованих систем рекомендацій.

3 ОПИС ПРОЄКТУ ПРОПОНОВАНОЇ СИСТЕМИ

Точне прогнозування рейтингів фільмів – це лише один аспект аналізованої системи рекомендацій світового класу. У цій частині допису роботи детальніше розглянемо ширше покращену технологію персоналізації. Обговоримо деякі з наших поточних моделей, даних та підходів, яких ми дотримуємося, щоб керувати інноваціями та дослідженнями в цій галузі.

3.1 Зв'язок рейтингу та рекомендацій. Ранжування

Мета рекомендаційних систем – представити людині низку привабливих продуктів на вибір. Зазвичай це досягається шляхом вибору кількох товарів та їх сортування в порядку очікуваного задоволення (або корисності). Оскільки найпоширеніший спосіб представлення рекомендованих товарів та послуг – це певний список, наприклад, різні рядки на Netflix, нам потрібна відповідна модель ранжування, яка може використовувати широкий спектр інформації для визначення оптимального рейтингу товарів для кожного з наших учасників.

Якщо ви шукаєте функцію ранжування, яка оптимізує споживання, очевидною базовою лінією є популярність (рейтинг) товарів. Причина зрозуміла: в середньому учасник, найімовірніше, дивиться те, що дивляться більшість інших. Однак популярність є протилежністю персоналізації: вона призведе до однакового порядку товарів для кожного учасника. Таким чином, метою стає пошук персоналізованої функції ранжування, яка буде кращою за популярність товарів, щоб ми могли краще задовольнити потреби учасників з різними смаками.

Нагадаємо, що мета дослідження є створити та/або покращити алгоритм рекомендацій ігор та фільмів, в які кожен учасник, найімовірніше, гратиме та які йому сподобаються. Один очевидний спосіб підійти до цього – використовувати прогнозований учасником рейтинг кожного предмета як доповнення до популярності предмета. Використання прогнозованих рейтингів окремо як

функції ранжування може призвести до того, що будуть рекомендовані занадто нішеві або незнайомі елементи, а також може виключити предмети, які учасник хотів би переглянути, навіть якщо він може не оцінювати їх високо. Щоб компенсувати це, замість того, щоб використовувати лише популярність або прогнозований рейтинг окремо, ми хотіли б створити рейтинги, які збалансують обидва ці аспекти. На даний момент можна побудувати модель прогнозування рейтингу, використовуючи ці дві функції.

Існує багато способів побудови функції ранжування, починаючи від простих методів оцінювання до попарних уподобань та оптимізації по всьому рейтингу. Для ілюстрації почнемо з дуже простого підходу до оцінювання, обравши нашу функцію ранжування як лінійну комбінацію популярності та прогнозованого рейтингу. Це дає рівняння виду (3.1):

$$\text{rank}(u,v) = w1\ p(v) + w2\ r(u,v) + b, \tag{3.1}$$

де u = користувач,

v = відеоелемент,

p = популярність та

r = прогнозований рейтинг.

Це рівняння визначає двовимірний простір, подібний до того, що зображено нижче на рисунку 3.1.

Як тільки у нас є така функція, ми можемо пропустити набір відео через нашу функцію та відсортувати їх у порядку спадання відповідно до оцінки. Може бути цікаво, як можемо встановити ваги w1 та w2 у нашій моделі (зміщення b є постійним і, таким чином, не впливає на кінцевий порядок). Іншими словами, у нашій простій двовимірній моделі, як ми визначаємо, чи популярність більш чи менш важлива, ніж прогнозований рейтинг? Існує принаймні два можливі підходи до цього.

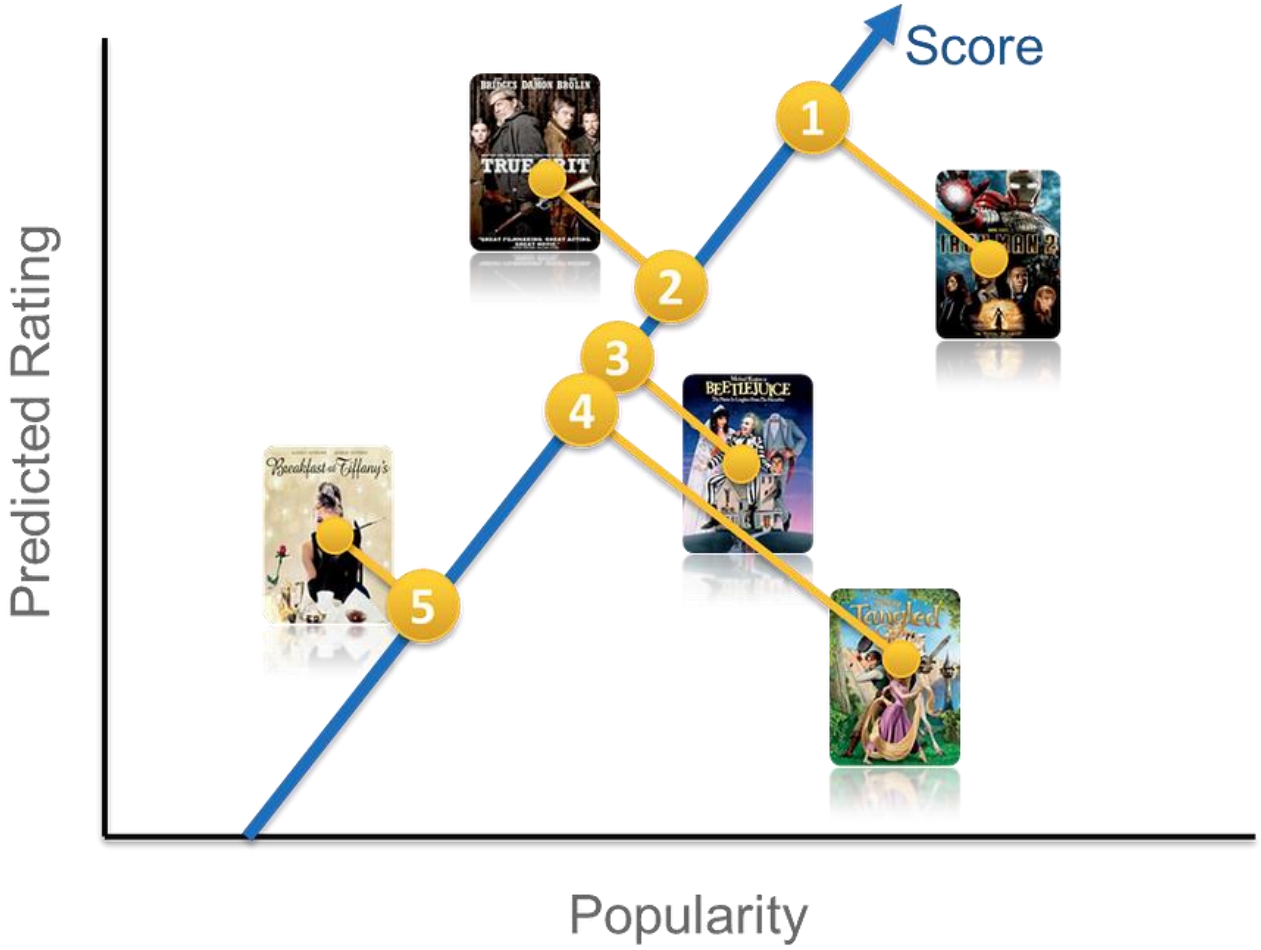


Рисунок 3.1 – Прості побудови рейтингів фільмів стрімінгової платформи

При розробці алгоритму рекомендацій можна вибрати простір можливих ваг і дозволити учасникам вирішити, що має сенс після багатьох A/B (сегментних) тестів. Ця процедура може бути трудомісткою та не дуже економічно ефективною. Інша можлива відповідь передбачає формулювання цього як задачі машинного навчання: виберіть позитивні та негативні приклади з ваших історичних даних і дозвольте алгоритму машинного навчання вивчити ваги, які оптимізують вашу мету. Це сімейство задач машинного навчання відоме як "Навчання ранжуванню (Learning to rank)" і є центральним для таких прикладних сценаріїв, як пошукові системи або таргетинг реклами. Варто зауважити, однак, що ключовою відмінністю у випадку рейтингових рекомендацій є важливість персоналізації: ми не очікуємо глобального поняття релевантності, а радше шукаємо способи оптимізації персоналізованої моделі.

Як можна здогадатися, окрім прогнозування популярності та рейтингу, було випробувано багато інших функцій Netflix. Деякі не показали жодного позитивного ефекту, тоді як інші значно покращили точність виконаного ранжування. Графік нижче (рисунок 3.2) показує покращення рейтингу, якого ми досягли завдяки додаванню різних функцій та оптимізації алгоритму машинного навчання.

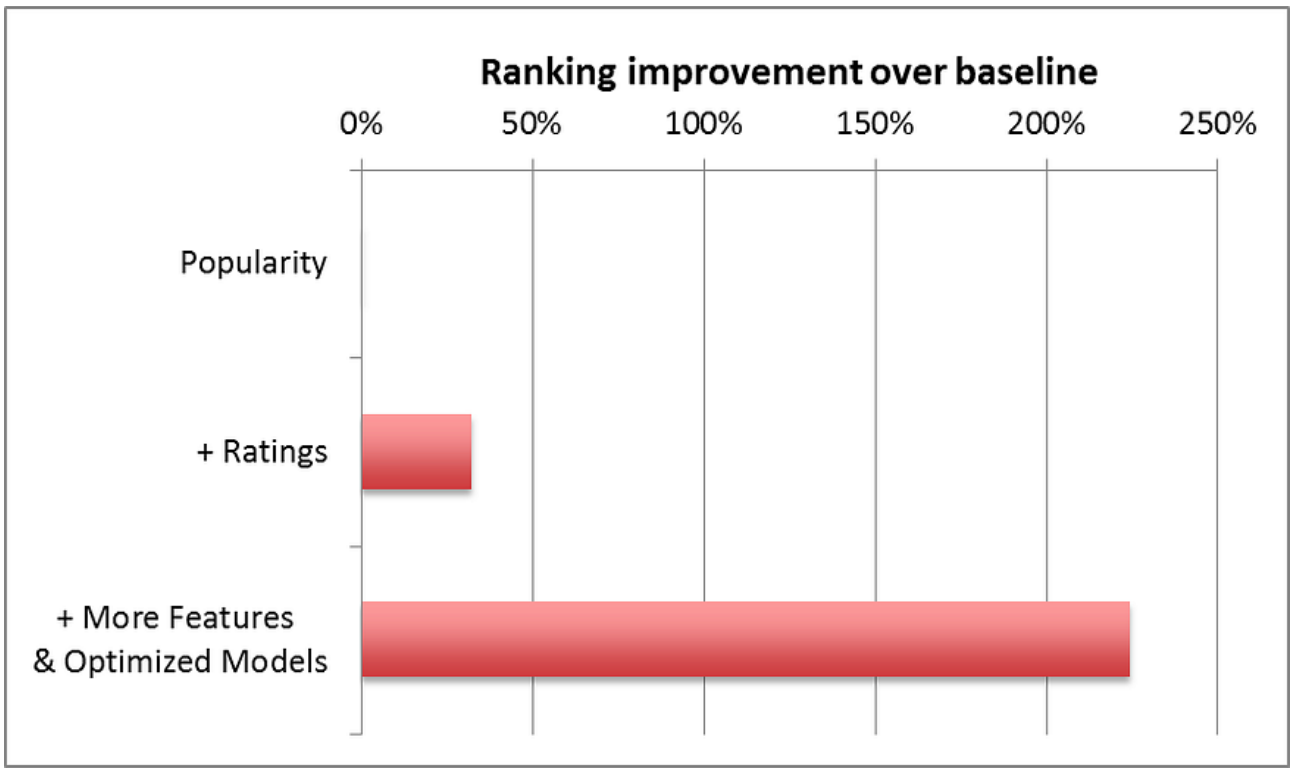


Рисунок 3.2 – Покращення рейтингу завдяки додаванню різних функцій та оптимізації алгоритму машинного навчання

Для ранжування можна використовувати багато методів контрольованої класифікації. Типовий вибір включає логістичну регресію, метод опорних векторів, нейронні мережі або методи на основі дерев рішень, такі як градієнтно-підсилені дерева рішень (GBDT). З іншого боку, в останні роки з'явилася велика кількість алгоритмів, спеціально розроблених для навчання ранжування, таких як RankSVM або RankBoost. Немає простої відповіді на питання, яка модель найкраще працюватиме в заданій задачі ранжування. Чим простіший простір ознак, тим простішою може бути вибрана модель. Але легко потрапити в пастку

ситуації, коли нова ознака не показує цінності, оскільки модель не може її вивчити. Або, навпаки, зробити висновок, що потужніша модель некорисна просто тому, що у вас немає простору ознак, який би використовував її переваги.

3.2 Дані взаємодії користувачів платформи та моделі побудови рекомендацій

У попередньому обговоренні алгоритмів ранжування було підкреслено важливість як даних, так і моделей для створення оптимального персоналізованого досвіду для користувачів сервісу. У Netflix наявні для застосування багато відповідних джерел даних та кваліфікованих сеціалістів, які можуть вибрати оптимальні алгоритми для перетворення даних на характеристики продукту. Ось деякі джерела даних, які ми можемо використовувати для оптимізації наших рекомендацій.

- Наявні кілька мільярдів оцінок від учасників. І мільйони нових оцінок отримуються щодня.
- Було вже згадування популярності товару як базового показника. Але існує багато способів обчислення популярності. Можна обчислювати її за різні часові діапазони, наприклад, щогодини, щодня або щотижня. Або ж ми можемо групувати учасників за регіоном чи іншими показниками подібності та обчислювати популярність у межах цієї групи.
- Щодня здійснюється кілька мільйонів відтворень стрімів, які включають такий контекст, як тривалість, час доби та тип пристрою.
- Щодня користувачі додають мільйони елементів до своїх черг.
- Кожен елемент у нашому каталозі має багато метаданих: актори, режисер, жанр, батьківський рейтинг та відгуки.
- Презентації. Відомо, які елементи були рекомендовані раніше та де їх показували, і можна побачити, як це рішення вплинуло на дії учасника. Також можна спостерігати за взаємодією учасника з рекомендаціями: прокручування, наведення курсору миші, кліки або час, проведений на певній сторінці.

- Соціальні дані (соцмережі) стали найновішим джерелом функцій персоналізації. Можна обробляти те, що переглянули або оцінили друзі, які нас підключили.
- Щодня безпосередньо вводять мільйони пошукових запитів у сервісі Netflix.
- Усі згадані вище дані надходять із внутрішніх джерел. Але також можна використовувати зовнішні дані для покращення функцій рекомендацій. Наприклад, ми можемо додавати зовнішні дані про продукти компанії, такі як касові збори або відгуки критиків.
- Звичайно, це ще не все – є багато інших характеристик, таких як демографічні дані, місцезнаходження, мова або часові дані, які можна використовувати в наших прогностичних моделях.

Одна річ, яку ми виявили в Netflix, полягає в тому, що з огляду на велику доступність даних, як за кількістю, так і за типами, потрібен продуманий підхід до вибору, навчання та тестування моделей. Ми використовуємо всілякі підходи до машинного навчання: від методів без учителя, таких як алгоритми кластеризації, до низки контрольованих класифікаторів, які показали оптимальні результати в різних контекстах. Це неповний список методів, про які слід враховувати в контексті вирішення задач чи побудови алгоритмів машинного навчання для персоналізації:

- Лінійна регресія.
- Логістична регресія.
- Еластичні сітки.
- Розкладання сингулярних значень.
- Граничні машини Больцмана.
- Ланцюги Маркова.
- Розподіл Діріхле.
- Правила асоціації.
- Дерева рішень з градієнтним посиленням.
- Випадкові ліси.

- Методи кластеризації від простих k-середніх до нових графічних підходів, таких як поширення спорідненості.
- Факторизація матриці.

3.3 Наука про дані від споживачів послуг

Велика кількість вихідних даних, вимірювань та пов’язаних з ними експериментів дозволяє управляти організацією, керованою даними. Netflix впровадив цей підхід у свою культуру з моменту заснування компанії. Загалом кажучи, головна мета підходу Netflix та досліджень вподобань клієнта (споживчої науки) – ефективне впровадження інновацій для усіх клієнтів. Єдина справжня невдача – це невдача в інноваціях; або, за словами Томаса Вотсона-старшого, засновника IBM: "Якщо ви хочете збільшити свій рівень успіху, подвоїте свій рівень невдач". Ми прагнемо до культури інновацій, яка дозволяє нам швидко, недорого та об’єктивно оцінювати ідеї. І, коли ми щось тестуємо, ми хочемо зрозуміти, чому воно зазнало невдачі чи успіху. Це дозволяє зосередитися на центральній меті – покращенні сервісу для надання якісних послуг споживачам цих послуг.

Отже, розглянемо, як це працює на практиці. Це невелика варіація від традиційного наукового процесу, який називається A/B-тестуванням (або пакетним тестуванням, коли всі користувачі розділяються на групи (пакети, кластери)):

1. Формулювання та висунення гіпотези.

- Алгоритм/функція/дизайн X підвищить залученість учасників до нашого сервісу та, зрештою, утримання учасників.

2. Розробка тестів.

- Розробіть рішення або прототип. Ідеальне виконання може бути вдвічі ефективнішим за прототип, але не в 10 разів.
- Подумайте про залежні та незалежні змінні, контроль, значущість.

3. Виконання тесту.

3.4 Дослідження даних та формування гіпотез

Коли ми проводимо A/B-тестування, ми відстежуємо багато різних показників. Але зрештою ми довіряємо залученості учасників (наприклад, годинам гри) та утриманню учасників. Тести зазвичай мають тисячі учасників і від 2 до 20 осередків, які досліджують варіації базової ідеї. Зазвичай паралельно працюють десятки A/B-тестів. A/B-тестування дозволяють випробувати радикальні ідеї або протестувати багато підходів одночасно, але ключова перевага полягає в тому, що вони дозволяють рішенням керуватися даними.

Цікаве подальше питання, з яким ми зіткнулися, полягає в тому, як інтегрувати підходи до машинного навчання в культуру A/B-тестування, керовану даними, в Netflix. Це пропонується виконати за допомогою процесу офлайн-тестування, який намагається поєднати найкраще з обох підходів. Цикл офлайн-тестування – це крок, на якому ми тестуємо та оптимізуємо алгоритми перед проведенням онлайн-A/B-тестування.

Щоб виміряти продуктивність моделі офлайн, ми відстежуємо кілька показників, що використовуються в спільноті машинного навчання: від показників ранжування, таких як нормалізований дисконтований кумулятивний приріст, середній зворотний ранг або частка конкордантних пар, до показників класифікації, таких як точність, прецизійність, повнота або F-оцінка. Ми також використовуємо відомий показник (RMSE) від Netflix Prize або інші більш екзотичні показники для відстеження різних аспектів, таких як різноманітність. Ми відстежуємо, наскільки добре ці показники корелюють з вимірюваними онлайн-приростами в наших A/B-тестах. Однак, оскільки зіставлення не є ідеальним, продуктивність офлайн використовується лише як показник для прийняття обґрунтованих рішень щодо подальших тестів (див. рис. 3.3).

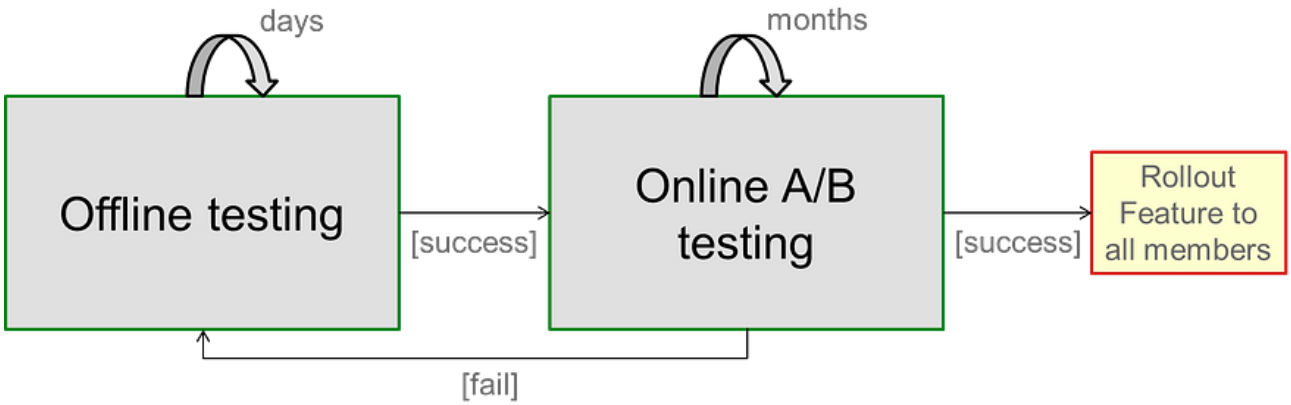


Рисунок 3.3 – Етапи сегментного тестування

Після того, як офлайн-тестування підтвердить гіпотезу, ми готові розробити та запустити A/B-тестування, яке доведе, що нова функція є придатною з точки зору учасників. Якщо це так, ми будемо готові розпочати нашу постійну роботу над покращенням продукту для наших учасників. Діаграма нижче (див. рис. 3.4) ілюструє деталі цього процесу.

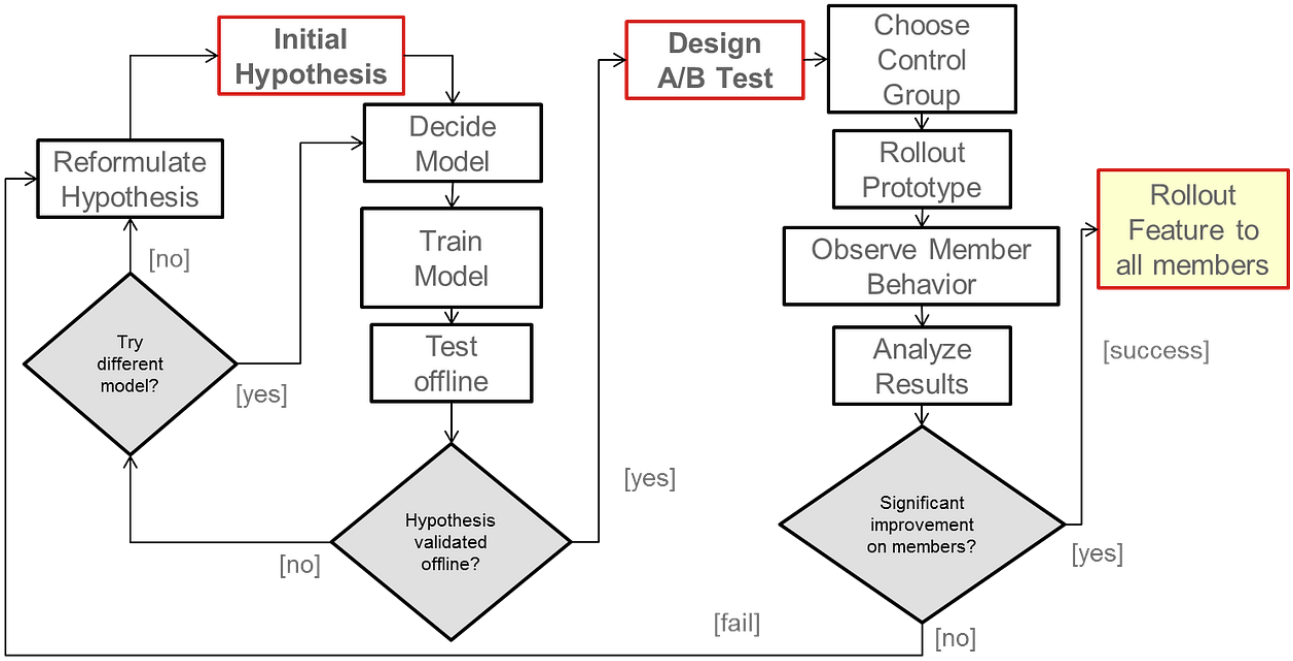


Рисунок 3.4 – Деталі впровадження нового алгоритму в рекомендаційну систему

Премія Netflix звела проблему рекомендацій до питання-посередника – прогнозування рейтингів. Але рейтинги учасників – це лише одне з багатьох

джерел даних, які ми маємо, а прогнози рейтингів – лише частина нашого рішення. З часом ми переформулювали проблему рекомендацій до питання оптимізації ймовірності того, що учасник вирішить переглянути певний фільм і йому сподобається він настільки, що він повернеться до сервісу. Більший доступ до даних забезпечує кращі результати. Але для того, щоб отримати ці результати, нам потрібні оптимізовані підходи, відповідні показники та швидке експериментування.

3.5 Розробка та тестування системи

Netflix прагне допомогти людям знайти улюблені фільми. Щоб допомогти клієнтам знаходити ці фільми, вони розробили систему рекомендацій фільмів найвищого класу CinematchSM. Її завдання полягає в тому, щоб передбачити, чи сподобається комусь фільм, на основі того, наскільки йому сподобалися чи не сподобалися інші фільми. Netflix використовує ці прогнози, щоб створювати персональні рекомендації фільмів на основі унікальних смаків кожного клієнта. І хоча Cinematch працює досить добре, його завжди можна покращити.

Зараз існує багато цікавих альтернативних підходів до роботи Cinematch, які Netflix ще не пробував. Деякі описані в літературі, деякі ні. Нам цікаво, чи може якийсь із них перевершити Cinematch, роблячи кращі прогнози. Тому що, чесно кажучи, якщо існує набагато кращий підхід, це може суттєво вплинути на наших клієнтів та наш бізнес.

Netflix надав багато анонімних даних про рейтинги та шкалу точності прогнозування, яка на 10% краща, ніж те, що може зробити Cinematch на тому ж наборі навчальних даних. (Точність – це вимірювання того, наскільки точно прогнозовані рейтинги фільмів відповідають наступним фактичним рейтингам).

Дані будемо брати з відкритого джерела [17].

Перший рядок кожного файлу даних містить ідентифікатор фільму, за яким йде двокрапка.

Кожен наступний рядок у файлі відповідає оцінці від клієнта та її даті у такому форматі:

```
CustomerID,Rating,Date5
```

Рейтинг фільму має значення від 1 до 5.

Зразок даних показано на рисунку 3.5.

```
1488844,3,2005-09-06
822109,5,2005-05-13
885013,4,2005-10-19
30878,4,2005-12-26
823519,3,2004-05-03
893988,3,2005-11-17
124105,4,2004-08-05
1248029,3,2004-04-22
1842128,4,2004-05-09
2238063,3,2005-05-11
1503895,4,2005-05-19
2207774,5,2005-06-06
2590061,3,2004-08-12
2442,3,2004-04-14
543865,4,2004-05-28
1209119,4,2004-03-23
804919,4,2004-06-10
1086807,3,2004-12-28
1711859,4,2005-05-08
372233,5,2005-11-23
1080361,3,2005-03-28
1245640,3,2005-12-19
558634,4,2004-12-14
2165002,4,2004-04-06
1181550,3,2004-02-01
1227322,4,2004-02-06
427928,4,2004-02-26
```

Рисунок 3.5 – Зразок даних про фільми

Для досягнення мети будемо використовувати метрики:

- Середня абсолютна відсоткова похибка.
- Середньоквадратична похибка.

Будемо мінімізувати середньоквадратичну похибку.

Після попереднього опрацювання даних. Усунення дублікатів, опрацювання пропущених значень ділимо весь набір даних на навчальну та тестову частини у традиційному співвідношення 80% (навчальна вибірка) на 20% (тестова вибірка).

Проаналізуємо розподіл значень рейтингів фільмів для навчально вибірки (рис. 3.6).

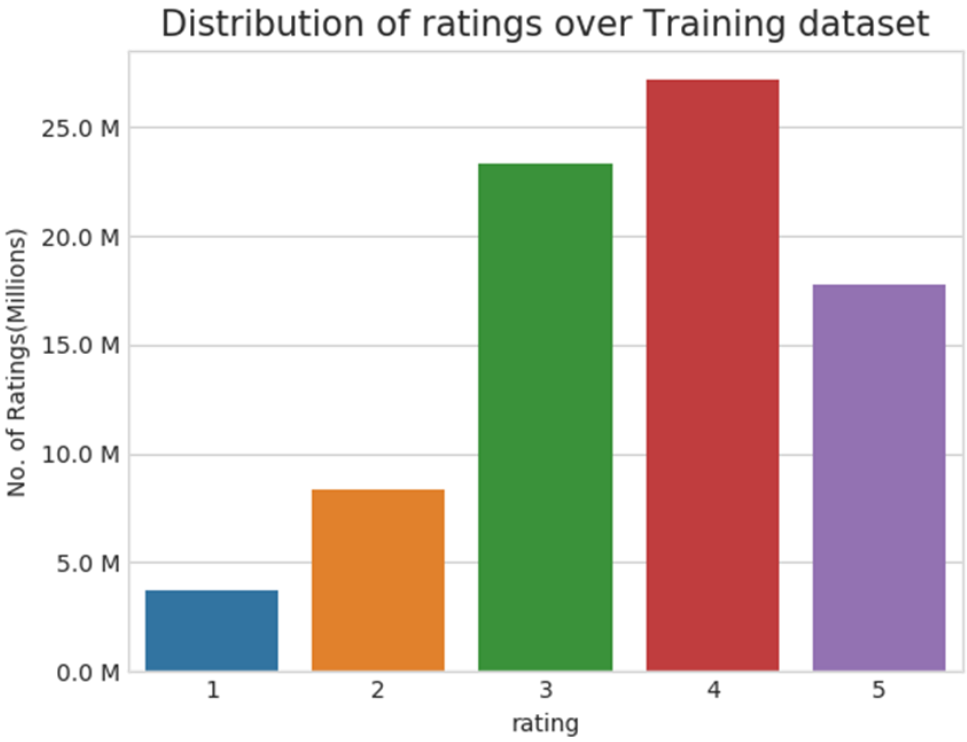


Рисунок 3.6 – Розподіл рейтингів фільмів навчальної вибірки

Також відобразимо фнукцію розподілу та щільність розподілу для значень рейтингів фільмів (див. рис. 3.7).

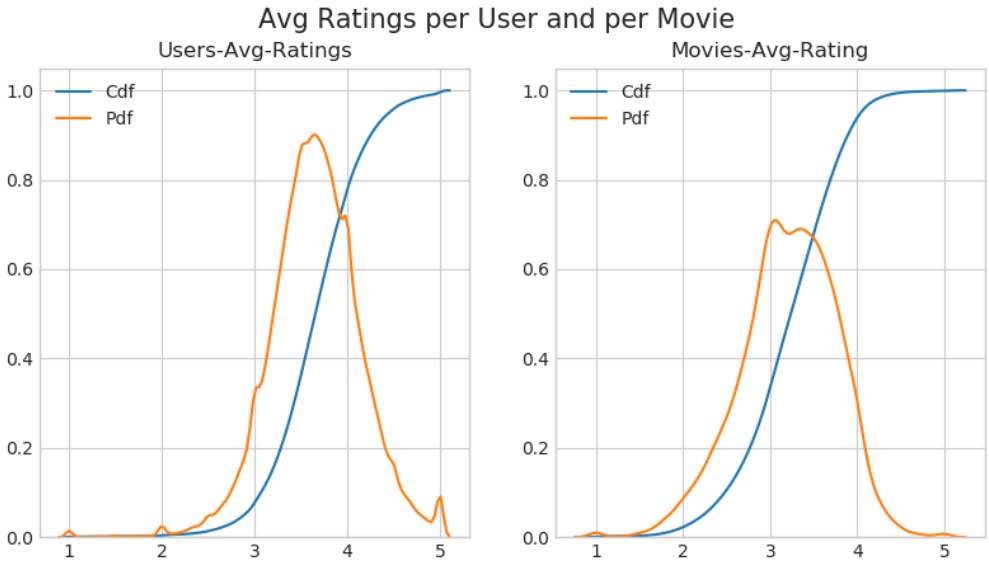


Рисунок 3.7 – Функція розподілу (CDF) та щільність розподілу (PDF) для значень рейтингів фільмів

Після опрацювання даних виведемо графік важливості (ваги) кожної характеристик набору даних (рисунок 3.8).

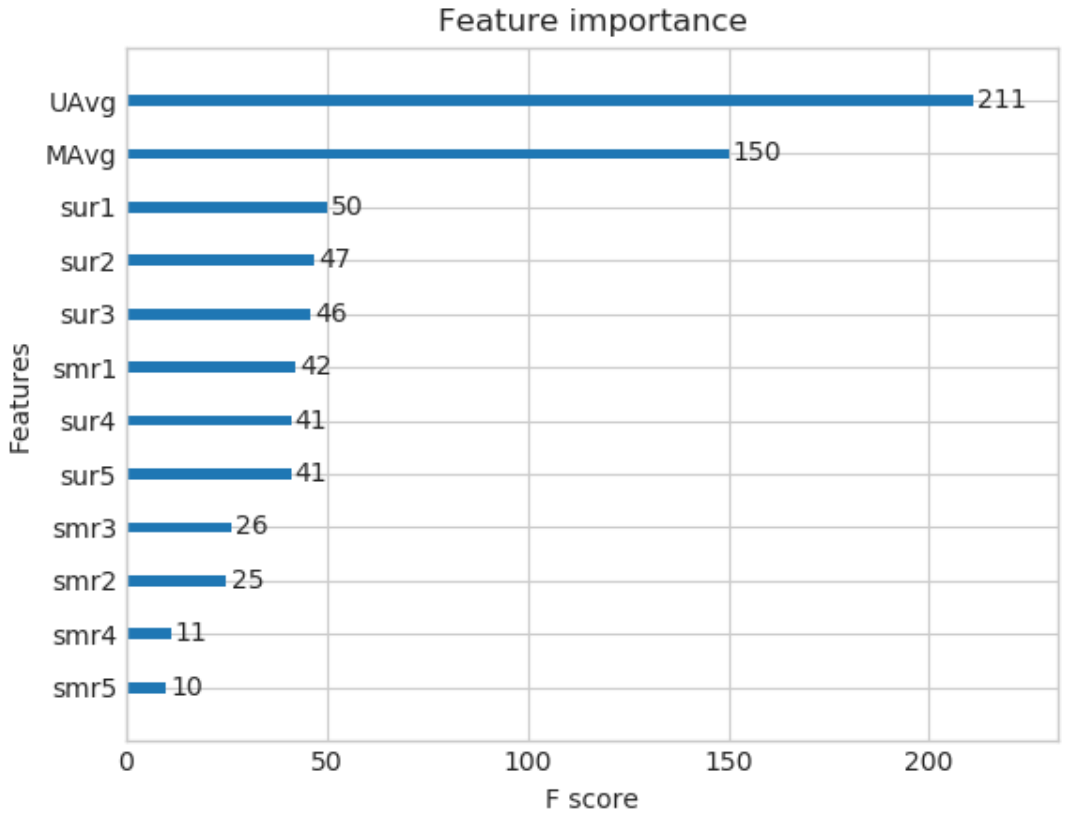


Рисунок 3.8 – Ваги характеристик набору даних

Було випробувано декілька моделей машинного навчання для побудови системи рекомендацій. Результати їх порівняння показано на рисунку 3.9.

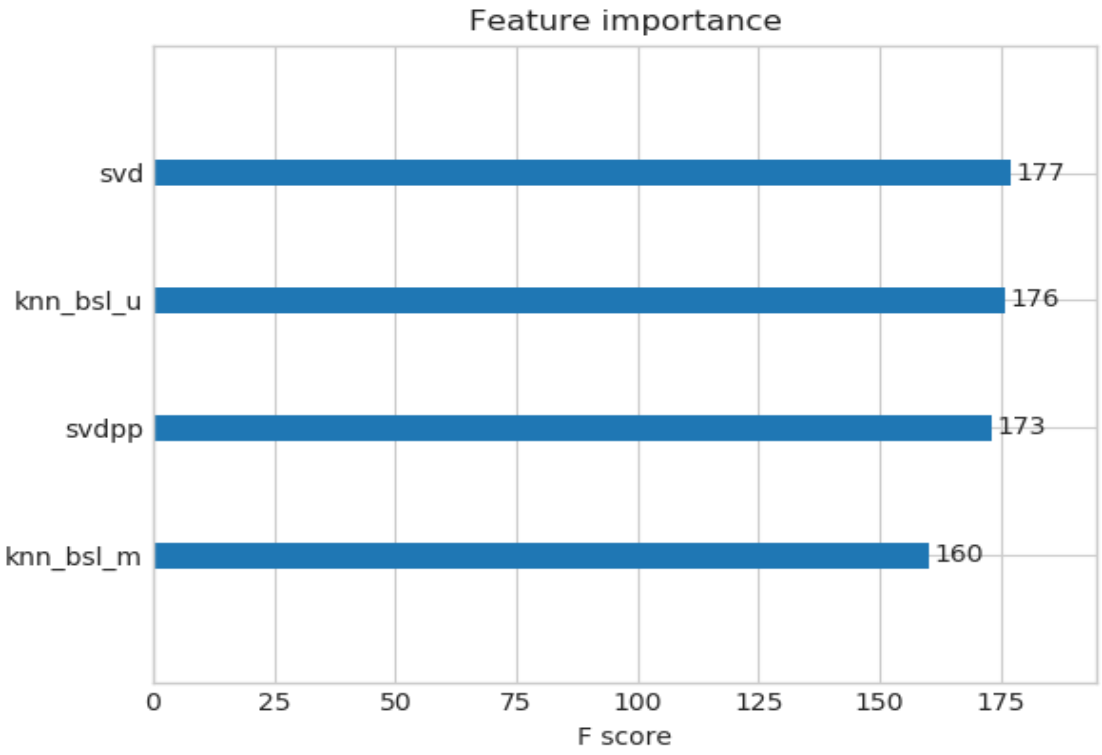


Рисунок 3.9 – Порівняння моделей машинного навчання для рекомендаційної системи

Пояснимо позначення на рисунку 3.9.

Svd – модель сингулярного розкладу для зменшення розмірності набору даних та для рекомендаційних систем.

Svdpp – (svd++) розширення класичного SVD, яке враховує не лише явні рейтинги користувачів, а й імпліцитну інформацію (наприклад, факт перегляду чи взаємодії з фільмом, навіть без оцінки).

І методи К найближчих сусідів (K nearest neighbors) для користувачів та фільмів:

Knn_bsl_u – user-based KNNBaseline (сусідство між користувачами).

Knn_bsl_m – item-based KNNBaseline (сусідство між елементами/фільмами).

Як бачимо, найкращі результати отримує метод Svd а також Knn_u.

Програмний код блокноту Jupyther Anaconda наведено в додатку до кваліфікаційної роботи.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В умовах прискореного розвитку інформаційних технологій, професії у сфері ІТ займають ключове місце в економічній структурі та соціальному устрої. Відповідно, набуває актуальності забезпечення адекватних умов праці та охорони здоров'я для спеціалістів цієї галузі. Попри загальноприйняту думку про порівняно низький ризик в роботі програмістів порівняно з іншими професіями, ця сфера містить специфічні ризики та особливості, що вимагають індивідуального підходу до питань охорони праці.

Аналіз робочого середовища ІТ-спеціалістів показує, що, незважаючи на зовнішню зручність, існують недоліки з точки зору ергономіки, психологічного навантаження та інших важливих аспектів. У цьому контексті важливим є вивчення основних аспектів охорони праці програмістів, аналіз потенційних ризиків та розробка рекомендацій щодо оптимізації умов праці для фахівців у галузі інформаційних технологій.

4.1 Питання щодо охорони праці

Фактори трудового середовища можуть істотно впливати на здоров'я та працездатність розробника програм. Неправильно організоване робоче місце може викликати проблеми із хребтом, шиєю, зап'ястям та іншими частинами тіла. Довгий час роботи за комп'ютером може призвести до тунельного синдрому зап'ястного каналу.

4.1.1 Ергономіка

У сучасних умовах трудової діяльності значуще місце займає організація робочого місця користувачів персональних комп'ютерів. Законодавство України наголошує на необхідності забезпечення безпечних та комфортних умов праці. Зокрема, відповідно до КЗпП та Закону України "Про охорону праці",

роботодавці мають обов'язки стосовно забезпечення працівників належними умовами.

При організації робочих місць із персональними комп'ютерами важливо забезпечити відстані між боковими частинами столів не менше ніж 1,2 м, а також враховувати, що відстань між задньою частиною одного комп'ютера та екраном іншого має бути 2,5 м. Структура робочого столу повинна бути розроблена відповідно до ергономічних стандартів, дозволяючи ефективно розташовувати необхідне обладнання, таке як дисплей, клавіатура, принтер, а також робочі документи.

Щодо параметрів робочого столу: його висота повинна знаходитися у діапазоні від 680 до 800 мм. Що стосується ширини та глибини, то вони повинні бути такими, щоб працівник міг з легкістю працювати в зоні доступності рук (для цього рекомендовані показники: ширини від 600 до 1400 мм, глибини від 800 до 1000 мм). Також необхідно передбачити комфортний простір для ніг користувача: висота – мінімум 600 мм, ширина – мінімум 500 мм, глибина на рівні колін – 450 мм і на рівні витягнутої ноги – не менше 650 мм.

Стілець на робочому місці повинен мати підйомно-поворотні характеристики, можливість регулювання по висоті, куту нахилу сидушки і спинки, а також відстані від спинки до зовнішнього краю сидіння. Поверхня сидіння має бути рівною, а її зовнішній край має мати округлу форму. Налаштування по кожному параметру повинно бути індивідуальним, інтуїтивним та надійно фіксуватися.

Інтервал регулювання частин стільця: для лінійних розмірів – 15-20 мм, для кутових 2-5°. Зусилля для регулювання не має перевищувати 20 Н. Висота сидіння повинна бути від 400 до 500 мм, а її ширина і глибина – не менше 400 мм. Сидіння має мати можливість нахилу до 15° вперед і до 5° назад. Висота спинки стільця – 300±20 мм, ширина – не менше 380 мм, радіус кривизни горизонтально – 400 мм. Кут нахилу спинки може регулюватися в діапазоні 1-30° від вертикального положення. Відстань від спинки до сидіння – від 260 до 400 мм. Для зменшення напруження в руках рекомендується використовувати

підлокітники довжиною від 250 мм, шириною 50-70 мм, що налаштовуються по висоті від 230 до 260 мм та по відстані між ними від 350 до 500 мм. Матеріал сидіння і спинки має бути напівтвердим, антиковзним, що пропускає повітря і легко миється, а також не збирає статичний заряд.

Робоча зона повинна бути обладнана підніжкою шириною мінімум 300 мм, глибиною не менше 400 мм, з можливістю регулювання по висоті до 150 мм і нахилом до 20°. На підніжці має бути рельєфна поверхня і маленький борт по зовнішньому краю висотою 10 мм. Екран комп'ютера слід розміщувати на оптимальній відстані від користувача, що варіюється в межах 500-700 мм, але не ближче ніж 500 мм, з урахуванням легкості читання тексту і зображень. Екран повинен бути розташований так, щоб забезпечити комфорт при спостереженні, кутом у 30° від вертикалі.

4.1.2 Освітлення

Ефективне та грамотне виробниче світло підвищує якість зорової діяльності, зменшує втомленість, стимулює зростання продуктивності, сприяє комфортному робочому середовищу, додаючи спокій та позитив працівникам, а також підсилює безпеку роботи, зменшуючи ризик травм. Недостатнє або надто яскраве освітлення може спричинити зорове перевтомлення і головний біль.

Недолік світла може призводити до перевантаження очей, зниження уваги та передчасної втоми. Занадто яскраве світло викликає сліпоту, незадоволення та відчуття дискомфорту в очах. Неправильне розташування світла може утворювати відблиски, контрастні тіні та дезорієнтувати працівника. Ці фактори можуть спричинити нещасний випадок або професійні захворювання, тому важливо правильно розраховувати інтенсивність світла.

Розрізняють три типи світла - природне, штучне та комбіноване.

Природне світло – це варіант освітлення, при якому денне світло проникає через вікна або інші прозорі елементи приміщення. Інтенсивність природного світла може сильно коливатися в залежності від доби, сезону та інших факторів.

Штучне світло використовують у вечірній час або коли природне світло недостатнє. Якщо природне світло доповнюється штучним, таке освітлення називають змішаним.

За своїм призначенням штучне світло може бути робочим, евакуаційним, аварійним чи охоронним. Робоче світло поділяється на загальне та місцеве. При загальному освітленні світильники розташовані рівномірно по приміщенню. У системі комбінованого освітлення до загального додається додаткове місцеве світло.

Згідно норм "ДБН В.2.5-28:2018 Природне і штучне освітлення", в комп'ютерних залах слід застосовувати комбінований тип освітлення. Для виконання робіт високої зорової точності (об'єкт розрізнення 0,3 ... 0,5 мм) інтенсивність природного світла має бути не менше 1,5%. Для робіт середньої точності (об'єкт розрізнення 0,5 ... 1,0 мм) - не менше 1,0%. Як джерела штучного світла часто використовують лампи типу ЛБ або ДРЛ, об'єднані в світильниках над робочими столами.

Вимоги до освітленості для робочих місць з комп'ютерами такі: при високій точності зорової роботи – 300 лк (загальне) та 750 лк (комбіноване); при середній точності – 200 та 300 лк відповідно.

Також важливо, щоб усе поле зору було якісно освітлене. Світлова інтенсивність приміщення та яскравість екрану комп'ютера повинні бути близькими, адже надлишкове світло може збільшувати втомленість очей.

4.1.3 Параметри мікроклімату

Для створення комфортних умов праці в приміщеннях з комп'ютерною технікою важливо дотримуватися певних стандартів мікроклімату. В таблиці 4.1 наведені параметри мікроклімату для таких приміщень

Таблиця 4.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Параметри мікроклімату	Літній період	Зимовий період
Температура повітря, °С	20 – 24	22 – 24
Відносна вологість, %	40 – 60	40 – 60
Швидкість руху повітря, м/с	0,1 – 0,2	0,1 – 0,2
Рівень шуму, дБ	до 50	До 50

4.1.4 Емоційна психогігієна

У сфері ІТ охорона праці включає не лише фізичне, але й психічне здоров'я працівників. Фактори, що впливають на психіку працівників у галузі інформаційних технологій, є різноманітними та мають велике значення у створенні здорового робочого середовища.

Таблиця 4.2 – Допустимі значення параметрів неіонізуючого електромагнітного випромінювання

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50см від поверхні монітора	10 В / м
Напруженість магнітної складової електромагнітного поля на відстані 50см від поверхні монітора	0,3 А / м
Напруженість електростатичного поля	20кВ / м

До них належать:

– тривалість та інтенсивність робочого дня: довгі години роботи без відпочинку можуть призвести до перевтоми та стресу, впливаючи на психічне здоров'я;

– терміни виконання завдань і робочий тиск: нереалістичні терміни виконання завдань або надмірний робочий тиск можуть викликати стрес та тривогу;

– міжособистісні взаємодії: конфлікти в команді або нездорова робоча атмосфера можуть спричиняти емоційне вигорання;

– робота в умовах ізоляції або віддалена робота: відсутність соціальної взаємодії або підтримки може вплинути на почуття ізоляції та самотності;

– work-life баланс : нездатність знайти баланс між роботою та особистим життям може призвести до стресу та емоційного вигорання;

– значне розумове навантаження: постійна потреба в освоєнні нових технологій та адаптації до змін може бути стресовою;

– відсутність автономії або контролю над роботою: обмежений контроль над робочими процесами та відсутність автономії можуть викликати почуття безпорадності та фрустрації.

Для ефективної профілактики психоемоційних проблем, пов'язаних з роботою в ІТ-галузі, необхідно розробити комплексний підхід, який враховує різноманітність факторів, що впливають на психічне здоров'я працівників. Значущість цього підходу полягає в його спрямованості на зменшення стресорів у робочому середовищі та підвищення ресурсів для психічного відновлення.

Першочергово, акцентується увага на регулюванні тривалості робочого дня та інтенсивності роботи. Це включає розробку ефективних графіків роботи, що передбачають достатні перерви для відновлення, та уникнення перевантаження завданнями. Крім того, розглядається важливість балансу між роботою та особистим життям, який може бути підтриманий через гнучкі робочі години та можливість дистанційної роботи.

Ще одним важливим аспектом є оптимізація робочого середовища з точки зору ергономіки. Покращення умов роботи на робочому місці, включаючи забезпечення якісного обладнання та комфортних умов, сприятиме зниженню фізичного та психологічного дискомфорту.

Ключову роль відіграє також розвиток корпоративної культури, яка підтримує психологічне благополуччя. Це передбачає створення відкритого, підтримуючого співробітництва та комунікації середовища, заохочення взаємодопомоги між колегами, та реалізацію програм з управління стресом.

Для забезпечення довгострокового психічного благополуччя, розглядається імплементація регулярних тренінгів та навчальних програм, спрямованих на розвиток навичок управління стресом та підвищення особистісної стійкості. Також важливим є застосування систематичного моніторингу психічного стану працівників з використанням психометричних інструментів для раннього виявлення потенційних проблем.

Розуміння та належне врахування цих факторів є ключовим для забезпечення психічного благополуччя працівників у галузі ІТ, що, у свою чергу, позитивно впливає на їх продуктивність та загальне задоволення роботою.

4.2 Питання щодо безпеки в надзвичайних ситуаціях

В умовах швидкого розвитку інформаційних технологій та зростання залежності бізнес-процесів від ІТ-систем, питання безпеки в сфері ІТ набуває особливої актуальності. Розглядаючи безпеку праці в ІТ, особливу увагу необхідно приділити розробці та імплементації процедур реагування на надзвичайні ситуації, що включають технічні збої, кібератаки, витік конфіденційної інформації, фізичні загрози обладнанню та перебої в електропостачанні, а також інші аварійні стани, здатні порушити звичний хід робочих процесів.

Відповідальність за реалізацію ефективної системи безпеки лежить на ІТ-спеціалістах, керівництві компаній та інших зацікавлених сторонах. Необхідно забезпечити встановлення чітких правил та інструкцій, які регламентують дії персоналу у випадку виникнення надзвичайних ситуацій. Це включає визначення швидких комунікаційних каналів, розробку планів відновлення

роботи систем, а також проведення регулярних тренінгів та інструктажів з охорони праці для всіх працівників.

Плани евакуації та реагування мають бути адаптовані до можливих IT-ризиків. Вони повинні включати дії для захисту життя та здоров'я працівників, а також процедури збереження даних та обладнання. Плани мають бути детальними та зрозумілими для кожного члена команди, з чіткими інструкціями щодо дій в кризових ситуаціях.

Регулярні тренінги з охорони праці, які охоплюють надзвичайні ситуації в IT, є необхідними для підготовки персоналу до ефективного реагування на інциденти. Навчальні програми повинні включати імітації надзвичайних ситуацій, навчання з використанням спеціалізованого обладнання та вивчення найкращих практик у сфері кібербезпеки. Систематичний аналіз ризиків є важливим для ідентифікації потенційних вразливостей системи та розробки відповідних заходів запобігання. Встановлення систем моніторингу, які можуть виявити ознаки надзвичайних ситуацій на ранніх стадіях, допомагає зменшити потенційні збитки та забезпечити оперативне реагування.

Після кожної надзвичайної ситуації необхідно проводити детальний аналіз її причин, ефективності дій персоналу та наявних процедур реагування. На основі цього аналізу слід вносити корективи у плани надзвичайних дій, поліпшувати процедури безпеки та організовувати додаткове навчання. Ефективне управління надзвичайними ситуаціями в IT-сфері вимагає всебічного підходу, який поєднує в собі як технічні, так і організаційні аспекти. Відповідальність за це лежить на всіх рівнях організації, починаючи з IT-фахівців і закінчуючи вищим керівництвом. Завдяки встановленню та дотриманню відповідних процедур можна мінімізувати ризики та забезпечити безперебійну роботу в умовах, що постійно змінюються.

ВИСНОВКИ

Щоб досягти успіху в інноваціях персоналізації, недостатньо бути методичним у наших дослідженнях; простір для дослідження практично безмежний. Ля кожної стрімінгової системи важливо зосередити дослідження на плідних напрямках для розвитку; недостатньо використані джерела даних, краще представлення функцій, більш відповідні моделі та метрики, а також втрачені можливості для персоналізації. Використовуються інтелектуальний аналіз даних та інші експериментальні підходи, щоб поступово інформувати нашу інтуїцію, а отже, пріоритезувати інвестування зусиль.

У більшості попередніх контекстів – будь то рядок Топ-10, жанри чи подібні – ранжування, вибір порядку розміщення елементів у рядку, є критично важливим для забезпечення ефективного персоналізованого досвіду. Мета даної системи ранжування – знайти найкращий можливий порядок набору елементів для учасника, в певному контексті, в режимі реального часу. Ми розкладаємо ранжування на оцінювання, сортування та фільтрацію наборів фільмів для показу учаснику. Бізнес-мета – максимізувати задоволеність учасників та щомісячне утримання підписки, що добре корелює з максимізацією споживання відеоконтенту. Тому ми оптимізуємо наші алгоритми, щоб давати найвищі оцінки тим програмам і фільмам, які учасник, найімовірніше, буде грати чи дивитись та які йому сподобаються.

Також потрібно враховувати такі фактори, як контекст, популярність фільму, інтерес, новизна, різноманітність та свіжість. Підтримка всіх різних контекстів, у яких ми хочемо надавати рекомендації, вимагає низки алгоритмів, налаштованих на потреби цих контекстів. У наступній частині цієї роботи детальніше розглянемо проблему ранжування. Також заглибимося в аналіз даних та моделі, які роблять все вищезазначене можливим, та обговоримо підхід до інновацій у цій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шумова, Л. О., Рязанцев, О. І., & Покришка, С. А. (2023). Моделі машинного навчання для формування рекомендацій. Вісник Східноукраїнського національного університету імені Володимира Даля, (2 (278)), 96-105.

2. ШЕВЧЕНКО, С., ЖДАНОВА, Ю., & ДАНИЛЮК, О. (2024). АНАЛІЗ ТА ДОСЛІДЖЕННЯ ХАРАКТЕРИСТИК АЛГОРИТМІВ У РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ. Вісник Херсонського національного технічного університету, (4 (91)), 367-376.

3. Пасічник, В. В., Юнчик, В. Л., Кунанець, Н. Е., & Федонюк, А. А. (2022). Використання нечіткої логіки у процесі експертного оцінювання електронних навчальних ресурсів. Scientific Bulletin of UNFU, 32(4), 66-76.

4. Pankiv, Y., Kunanets, N., Artemenko, O., Veretennikova, N., & Nebesnyi, R. (2021, September). Project of an intelligent recommender system for parking vehicles in smart cities. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 419-422). IEEE.

5. Matsyuk, O., Nazaruk, M., Turbal, Y., Veretennikova, N., & Nebesnyi, R. (2018, September). Information analysis of procedures for choosing a future specialty. In Conference on Computer Science and Information Technologies (pp. 364-375). Cham: Springer International Publishing.

6. Небесний, Р. М. (2023). Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями (Doctoral dissertation, Тернопільський національний технічний університет імені Івана Пулюя).

7. Ржеуський, А., Кунанець, Н., & Стахів, М. (2018). Рекомендаційна система інформаційного обслуговування користувачів бібліотек. У Матеріали V науково-технічної конференції "Інформаційні моделі, системи та технології" (с. 37). Тернопіль: ТНТУ.

8. Hsiao Mark (Ko-Jen). From Stranger Things to Your Favorite Things: Netflix's Recommendation Evolution. VideoRecSys 2024: Large-Scale Video

Recommender Systems Workshop. URL: https://videorecsys.com/slides/mark_talk3.pdf (date of access: 08.10.2025).

9. “Attention Is All You Need,” Ashish Vaswani et al., Proceedings of the 31st International Conference on Neural Information Processing Systems, arXiv:1706.03762v7, revised on 2 August 2023.

10. Joshi, S., Feng, Y., Hsiao, K. J., Zhang, Z., & Lamkhede, S. (2024, October). Sliding Window Training-Utilizing Historical Recommender Systems Data for Foundation Models. In Proceedings of the 18th ACM Conference on Recommender Systems (pp. 835-837).

11. KV Caching Explained: Optimizing Transformer Inference Efficiency. Hugging Face – The AI community building the future. URL: https://huggingface.co/blog/not-lain/kv-caching (date of access: 14.11.2025).

12. Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., & Basilico, J. (2021). Deep learning for recommender systems: A Netflix case study. AI magazine, 42(3), 7-18.

13. C. K. Kang and J. McAuley, “Self-Attentive Sequential Recommendation,” 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 2018, pp. 197–206, doi: 10.1109/ICDM.2018.00035.

14. F. Sun et al., “BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer,” Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM ‘19), Beijing, China, 2019, pp. 1441–1450, doi: 10.1145/3357384.3357895.

15. J. Zhai et al., “Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations,” arXiv preprint arXiv:2402.17152, 2024.

16. F. Gloeckle, B. Youbi Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve, “Better & Faster Large Language Models via Multi-token Prediction,” arXiv preprint arXiv:2404.19737, Apr. 2024.

17. Netflix Prize data. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/netflix-inc/netflix-prize-data/data (date of access: 08.05.2025).

18. Голінько В. І. Охорона праці в галузі інформаційних технологій: навч. посіб. / В. І. Голінько, М. Ю. Іконніков, Я. Я. Лебедєв; М-во освіти і науки України, Держ. вищий навч. закл. "Нац. гірн. ун-т". - Дніпропетровськ: НГУ, 2015. - 246 с.

19. Гандзюк М.П. Основи охорони праці: Підручник. 4-е вид./Гандзюк М.П., Желібо Є.П., Халімовський М.О. - Київ: Каревела, 2008. – 384с.

20. Техноекологія та цивільна безпека. Частина «Цивільна безпека»: Навчальний посібник; укл.: Стручок В. С. Тернопіль: ФОП Паляниця В.А., 2022. 150 с.

21. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

22. Умови праці працівників, які використовують у роботі персональні комп'ютери. Zolochiv.Net. URL: https://zolochiv.net/umovy-pratsi-pratsivnykiv-iaki-vykorystovuiut-u-roboti-personal-ni-komp-iutery/ (дата звернення: 25.10.2024).

ДОДАТКИ

Додаток А

Тези доповіді

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

УДК 004.451:004.738.5:004.75
В.А. Яцишин, П.В. Налутка, О.В. Доберчак, І.О. Боднарчук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

СУЧАСНА АРХІТЕКТУРА СТРІМІНГОВИХ ПЛАТФОРМ В РЕАЛЬНОМУ ЧАСІ

V.A. Yatsyshyn, P.V. Nalutka, O.V. Doberchak, I.O. Bodnarchuk

MODERN ARCHITECTURE OF REAL-TIME STREAMING PLATFORMS

На відміну від систем минулих часів, які спиралися на асинхронну та пакетну обробку, архітектура програмного забезпечення реального часу зараз є важливою в сучасному цифровому світі, де миттєва обробка інформації є нормою. Швидкість, надійність та передбачуваність є ключовими в застосунках реального часу, від платформ електронної комерції, що реагують на дії клієнтів, до фінансових систем, що виконують транзакції за мікросекунди. Саме тут на допомогу приходить архітектура програмного забезпечення реального часу, яка робить рівень продуктивності цих застосунків прийнятним для роботи в режимі реального часу.

Термін «реальний час» означає для системи, що вона повинна реагувати в чітко встановлені терміни або дедлайни. Для архітекторів та розробників розуміння того, як проектувати ці системи, є критично важливим для створення рішень у відповідності од сформульованих вимог. Навіть правильний результат може бути марним, якщо він надходить занадто пізно.

Системи реального часу поділяються на три категорії:

- Системи жорсткого реального часу. Недотримання термінів є системним збоєм. Прикладами є промислові системи управління та торговельні платформи, де час транзакцій є критично важливим.
- Системи роботи в режимі реального часу надійних систем. Пропуск термінів погіршує якість обслуговування, але не призводить до збою системи. Прикладами є відеоконференції, де випадкові переривання кадрів дратують, але не завершують дзвінок.
- Системи м’якого реального часу. Корисність результату знижується після закінчення терміну, але система продовжує функціонувати. Прикладами є механізми рекомендацій контенту, де персоналізація з невеликою затримкою все ще цінна.

Щоб врахувати ці різні очікування, цільовий додаток має бути спроектований та розроблений з урахуванням продуктивності в режимі реального часу. Незалежно від очікуваних термінів отримання результатів, добре побудована архітектура реального часу керує ресурсами, плануванням завдань, зв'язком та обробкою помилок, щоб забезпечити дотримання часових обмежень для своєї конкретної категорії.

Також важливо розуміти, що підпадає під архітектуру програмного забезпечення реального часу, а що ні (таблиця 1). Пакетна обробка, яка опрацьовує дані масово через заплановані проміжки часу, є класичним прикладом системи, що не працює в реальному часі. Хоча архітектура реального часу є поширеним шляхом, не всі варіанти використання та сценарії вимагають таких можливостей.

Майже всі програми містять компоненти реального часу поряд з аналізом історичних даних. Цей зсув підкреслює важливу роль програмного забезпечення та архітектур даних реального часу в сучасних програмах, що зумовлено ключовими факторами, серед яких варто згадати наступні.

Операції, критично важливі для бізнесу.

Для багатьох систем час впливає на бізнес-результати. Численні програми та галузі покладаються на справжні системи реального часу, щоб забезпечити дохід. Ось деякі приклади цього:

- Платформи електронної комерції: оновлення товарних запасів у режимі реального часу, персоналізація та обробка транзакцій безпосередньо впливають на коефіцієнти конверсії та задоволеність клієнтів.
- Фінансові послуги: торговельні платформи, системи обробки платежів та виявлення шахрайства потребують для роботи часу реагування на рівні мілісекунд.

Таблиця 1. Порівняння програмної архітектури реального часу та решти

Аспект	**Архітектура програмного забезпечення реального часу**	**Архітектура програмного забезпечення не для роботи в реальному часі**
випадки використання	Виявлення шахрайства, персоналізація в режимі реального часу, моніторинг у реальному часі, торгові додатки	Нарахування заробітної плати, створення звітів, пакетний імпорт, публікація контенту
Критерії успіху	Залежить як від точності , *так і* від своєчасності результатів	Залежить виключно від точності, незалежно від того, коли буде отримано результат
Дизайн застосунку	Компоненти, що реагують на події, неблокуючі та враховують час	Синхронні, запит/відповідь, блокувальні потоки прийнятні
Структура коду	Пріоритетність передбачуваних шляхів виконання, мінімальний вплив на збирання сміття (GC), асинхронний ввід/вивід	Надає пріоритет ремонтопридатності або пропускній здатності над точністю часу
Вплив технічного боргу	Архітектурно-технічний борг створює затримки, непередбачуваність та пропущені терміни — катастрофічні наслідки для систем реального часу. Навіть незначні недоліки, такі як блокування викликів або необмежені черги, можуть порушувати угоди про рівень обслуговування (SLA) або спричиняти каскадні збої.	Борг уповільнює доставку, збільшує витрати на обслуговування та може погіршити продуктивність, але рідко призводить до негайного збою. Терміни є гнучкими.

Кращий користувацький досвід.

Як ми вже обговорювали, очікування «миттєвого» обслуговування є складним завданням. Справжнього миттєвого зворотного зв'язку можна досягти лише завдяки інтеграції можливостей реального часу в базові сервіси. Наприклад, користувачі очікують миттєвого зворотного зв'язку, використовуючи:

- Веб- та мобільні додатки: адаптивні інтерфейси із часом завантаження менше секунди та миттєвими оновленнями (наприклад, стрічки соціальних мереж, спільне редагування) зараз є нормою [3].
- Потокові сервіси: доставка контенту з мінімальною буферизацією та адаптивною якістю вимагає прийняття рішень у режимі реального часу.

- Резервні механізми зі зниженою, але прийнятною продуктивністю.
- Моніторинг справності з швидким виявленням збоїв.

5. Моделі узгодженості даних.

Залежно від типу даних та рішень, що отримуються на їх основі, багато систем реального часу послаблюють сувору узгодженість для підвищення продуктивності. У таких випадках зазвичай застосовують:

- Узгоджені моделі для некритичних даних.
- Стратегії вирішення конфліктів для одночасних оновлень для підтримки цілісності даних.
- Шаблони CQRS (Command Query Responsibility Segregation) для розділення операцій читання та запису.

6. Подієво-орієнтований дизайн

Асинхронні, подієво-орієнтовані архітектури часто формують основу систем реального часу. Це означає, що кодові та архітектурні компоненти системи включатимуть:

- Брокери повідомлень, такі як Kafka або RabbitMQ, для надійної та впорядкованої доставки подій [2].
- Шаблони джерел подій для змін стану, що підлягає аудиту.
- Потокова обробка для безперервного аналізу даних.

Дотримуючись цих принципів, розробники можуть створювати системи, які відповідають потребам реального часу їхніх випадків використання. Ці шість принципів складають основні вимоги під час проектування та впровадження програм і сервісів реального часу. Крім того, розуміння різних аспектів продуктивності має вирішальне значення для системи реального часу. Це буде предметом нашого наступного обговорення.

Архітектура програмного забезпечення реального часу еволюціонувала від нішевої потреби у вбудованих системах до загальноприйнятого підходу для сучасних додатків. Оскільки компанії прагнуть надавати досвід, керований даними, здатність обробляти дані та реагувати в режимі реального часу стала ключовою конкурентною перевагою.

У цій доповіді досліджувалися основи систем реального часу: їх класифікація (жорсткі, стійкі, м'які), керівні принципи та ключові показники продуктивності.

Сучасні архітектури реального часу спираються на такі технології, як платформи потокової передачі подій (Kafka), сховища даних в пам'яті, моделі реактивного програмування та хмарні шаблони. При продуманому поєднанні ці компоненти дозволяють створювати масштабовані системи, які відповідають суворим гарантіям часу. Але чудове програмне забезпечення – це не лише технологія, а й те, як воно архітектурно спроектовано.

Щоб задовольнити вимоги систем реального часу, архітекторам потрібна постійна видимість того, як створюються та працюють додатки.

Література

1. Microservices architecture and design: A complete overview - vFunction. vFunction. URL: https://vfunction.com/blog/microservices-architecture-guide/ (date of access: 01.10.2025).

2. Ланевич Т. Apache Kafka та Rabbitmq: порівняння архітектур та можливостей // Тези XIII МНПК „Актуальні задачі сучасних технологій“, 11-12 грудня 2024 року. Тернопіль: ФОП Паляниця В. А., 2024. С. 402–403.

3. Хом’як А. С. Підвищення ефективності обробки в реальному часі у службах чат-ботів через інтеграцію черги запитів для розподілення навантаження //

Додаток Б

Програмний код Python фрагментарно

2. MACHINE LEARNING PROBLEM

2.1 Data

2.1.1 Data Overview

Get the data from : https://www.kaggle.com/netflix-inc/netflix-prize-data/data

2.1.2 Example Data point

```
1:
2590061,3,2004-08-12
2442,3,2004-04-14
543865,4,2004-05-28
1209119,4,2004-03-23
804919,4,2004-06-10
1086807,3,2004-12-28
1711859,4,2005-05-08
372233,5,2005-11-23
1080361,3,2005-03-28
1245640,3,2005-12-19
558634,4,2004-12-14
2165002,4,2004-04-06
1181550,3,2004-02-01
1227322,4,2004-02-06
427928,4,2004-02-26
814701,5,2005-09-29
808731,4,2005-10-31
662870,5,2005-08-24
337541,5,2005-03-23
786312,3,2004-11-16
1133214,4,2004-03-07
1537427,4,2004-03-29
```

2.2 Mapping the real world problem to a Machine Learning Problem

2.2.1 Type of Machine Learning Problem

```
For a given movie and user we need to predict the rating would be given by him/her to
the movie.
The given problem is a Recommendation problem
It can also seen as a Regression problem
```

2.2.2 Performance metric

2.2.3 Machine Learning Objective and Constraints

1. Minimize RMSE.
2. Try to provide some interpretability.

In [0]

```
# this is just to know how much time will it take to run this entire ipython notebook
from datetime import datetime
# globalstart = datetime.now()
import pandas as pd
import numpy as np
import matplotlib
matplotlib.use('nbagg')

import matplotlib.pyplot as plt
plt.rcParams.update({'figure.max_open_warning': 0})

import seaborn as sns
sns.set_style('whitegrid')
import os
from scipy import sparse
from scipy.sparse import csr_matrix

from sklearn.decomposition import TruncatedSVD
from sklearn.metrics.pairwise import cosine_similarity
import random
```

3. EXPLORATORY DATA ANALYSIS

3.1 Preprocessing

3.1.1 Converting / Merging whole data to required format: u_i, m_j, r_ij

In [0]

```
start = datetime.now()
if not os.path.isfile('data.csv'):
    # Create a file 'data.csv' before reading it
    # Read all the files in netflix and store them in one big file('data.csv')
    # We re reading from each of the four files and appendig each rating to a global
file 'train.csv'
    data = open('data.csv', mode='w')

    row = list()
    files=['data_folder/combined_data_1.txt','data_folder/combined_data_2.txt',
           'data_folder/combined_data_3.txt', 'data_folder/combined_data_4.txt']
    for file in files:
        print("Reading ratings from {}...".format(file))
        with open(file) as f:
            for line in f:
```

```
                del row[:] # you don't have to do this.
                line = line.strip()
                if line.endswith(':'):
                    # All below are ratings for this movie, until another movie
appears.
                    movie_id = line.replace(':', '')
                else:
                    row = [x for x in line.split(',')]
                    row.insert(0, movie_id)
                    data.write(','.join(row))
                    data.write('\n')
        print("Done.\n")
    data.close()
print('Time taken :', datetime.now() - start)

print("creating the dataframe from data.csv file..")
df = pd.read_csv('data.csv', sep=',',
                       names=['movie', 'user','rating','date'])
df.date = pd.to_datetime(df.date)
print('Done.\n')

# we are arranging the ratings according to time.
print('Sorting the dataframe by date..')
df.sort_values(by='date', inplace=True)
print('Done..')
```

Output (stdout)

```
creating the dataframe from data.csv file..
Done.

Sorting the dataframe by date..
Done..
```

In [0]

```
df.head()
```

HTML Output

	movie	user	rating	date
56431994	10341	510180	4	1999-11-11
9056171	1798	510180	5	1999-11-11
58698779	10774	510180	3	1999-11-11
48101611	8651	510180	2	1999-11-11
81893208	14660	510180	2	1999-11-11

Text Output

```
          movie    user  rating       date
56431994  10341  510180       4 1999-11-11
9056171    1798  510180       5 1999-11-11
58698779  10774  510180       3 1999-11-11
48101611   8651  510180       2 1999-11-11
81893208  14660  510180       2 1999-11-11
```

In [0]

```
df.describe()['rating']
```

Text Output

```
count    1.004805e+08
mean     3.604290e+00
std      1.085219e+00
min      1.000000e+00
25%      3.000000e+00
50%      4.000000e+00
75%      4.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

3.1.2 Checking for NaN values

In [0]

```
# just to make sure that all Nan containing rows are deleted..
print("No of Nan values in our dataframe : ", sum(df.isnull().any()))
```

Output (stdout)

```
No of Nan values in our dataframe :  0
```

3.1.3 Removing Duplicates

In [0]

```
dup_bool = df.duplicated(['movie','user','rating'])
dups = sum(dup_bool) # by considering all columns..( including timestamp)
print("There are {} duplicate rating entries in the data..".format(dups))
```

Output (stdout)

```
There are 0 duplicate rating entries in the data..
```

3.1.4 Basic Statistics (#Ratings, #Users, and #Movies)

In [0]

```
print("Total data ")
print("-"*50)
print("\nTotal no of ratings :",df.shape[0])
print("Total No of Users   :", len(np.unique(df.user)))
print("Total No of movies  :", len(np.unique(df.movie)))
```

Output (stdout)

```
Total data
--------------------------------------------------

Total no of ratings : 100480507
Total No of Users   : 480189
Total No of movies  : 17770
```

3.2 Spliting data into Train and Test(80:20)

In [0]

```
if not os.path.isfile('train.csv'):
    # create the dataframe and store it in the disk for offline purposes..
    df.iloc[:int(df.shape[0]*0.80)].to_csv("train.csv", index=False)

if not os.path.isfile('test.csv'):
    # create the dataframe and store it in the disk for offline purposes..
    df.iloc[int(df.shape[0]*0.80):].to_csv("test.csv", index=False)

train_df = pd.read_csv("train.csv", parse_dates=['date'])
test_df = pd.read_csv("test.csv")
```

3.2.1 Basic Statistics in Train data (#Ratings, #Users, and #Movies)

In [0]

```
# movies = train_df.movie.value_counts()
# users = train_df.user.value_counts()
print("Training data ")
print("-"*50)
print("\nTotal no of ratings :",train_df.shape[0])
print("Total No of Users   :", len(np.unique(train_df.user)))
print("Total No of movies  :", len(np.unique(train_df.movie)))
```

Output (stdout)

```
Training data
--------------------------------------------------

Total no of ratings : 80384405
Total No of Users   : 405041
Total No of movies  : 17424
```

3.2.2 Basic Statistics in Test data (#Ratings, #Users, and #Movies)

In [0]

```
print("Test data ")
print("-"*50)
print("\nTotal no of ratings :",test_df.shape[0])
print("Total No of Users   :", len(np.unique(test_df.user)))
print("Total No of movies  :", len(np.unique(test_df.movie)))
```

Output (stdout)

```
Test data
--------------------------------------------------

Total no of ratings : 20096102
Total No of Users   : 349312
Total No of movies  : 17757
```

3.3 Exploratory Data Analysis on Train data

In [0]

```
# method to make y-axis more readable
```

```
def human(num, units = 'M'):
    units = units.lower()
    num = float(num)
    if units == 'k':
        return str(num/10**3) + " K"
    elif units == 'm':
        return str(num/10**6) + " M"
    elif units == 'b':
        return str(num/10**9) +  " B"
```

3.3.1 Distribution of ratings

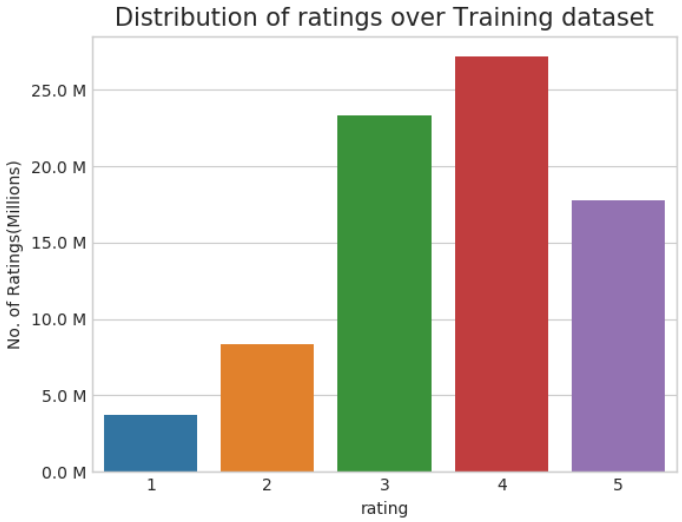
In [0]

```
fig, ax = plt.subplots()
plt.title('Distribution of ratings over Training dataset', fontsize=15)
sns.countplot(train_df.rating)
ax.set_yticklabels([human(item, 'M') for item in ax.get_yticks()])
ax.set_ylabel('No. of Ratings(Millions)')

plt.show()
```

Text Output

```
<IPython.core.display.Javascript object>
```

HTML Output



Text Output

```
<IPython.core.display.HTML object>
```

3.3.7.4 PDF's & CDF's of Avg.Ratings of Users & Movies (In Train Data)

In [0]

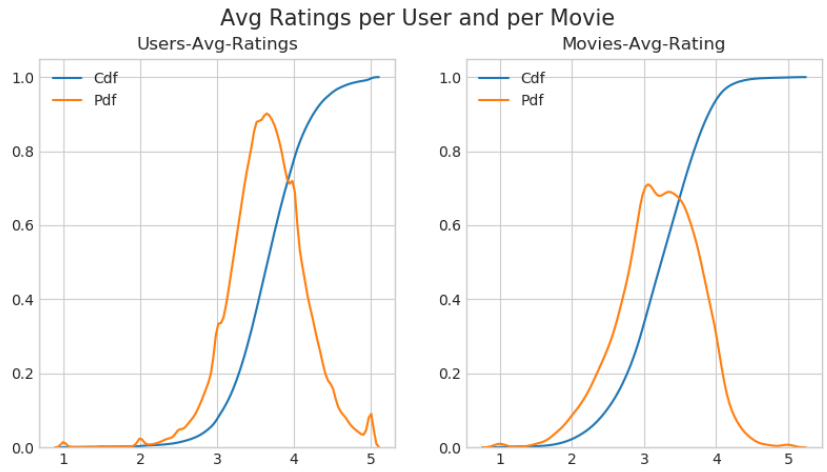
```
start = datetime.now()
# draw pdfs for average rating per user and average
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=plt.figaspect(.5))
fig.suptitle('Avg Ratings per User and per Movie', fontsize=15)

ax1.set_title('Users-Avg-Ratings')
```

```
# get the list of average user ratings from the averages dictionary..
user_averages = [rat for rat in train_averages['user'].values()]
sns.distplot(user_averages, ax=ax1, hist=False,
             kde_kws=dict(cumulative=True), label='Cdf')
sns.distplot(user_averages, ax=ax1, hist=False,label='Pdf')

ax2.set_title('Movies-Avg-Rating')
# get the list of movie_average_ratings from the dictionary..
movie_averages = [rat for rat in train_averages['movie'].values()]
sns.distplot(movie_averages, ax=ax2, hist=False,
             kde_kws=dict(cumulative=True), label='Cdf')
sns.distplot(movie_averages, ax=ax2, hist=False, label='Pdf')

plt.show()
print(datetime.now() - start)
```

Text Output

```
<IPython.core.display.Javascript object>
```

HTML Output



Text Output

```
<IPython.core.display.HTML object>
```

Output (stdout)

```
0:00:35.003443
```

3.4 Computing Similarity matrices

4. MACHINE LEARNING MODELS

In [0]

```
def get_sample_sparse_matrix(sparse_matrix, no_users, no_movies, path, verbose = True):
    """
        It will get it from the ''path'' if it is present  or It will create
        and store the sampled sparse matrix in the path specified.
```

```
    """

    # get (row, col) and (rating) tuple from sparse_matrix...
    row_ind, col_ind, ratings = sparse.find(sparse_matrix)
    users = np.unique(row_ind)
    movies = np.unique(col_ind)

    print("Original Matrix : (users, movies) -- ({} {})".format(len(users),
len(movies)))
    print("Original Matrix : Ratings -- {}\n".format(len(ratings)))

    # It just to make sure to get same sample everytime we run this program..
    # and pick without replacement....
    np.random.seed(15)
    sample_users = np.random.choice(users, no_users, replace=False)
    sample_movies = np.random.choice(movies, no_movies, replace=False)
    # get the boolean mask or these sampled_items in originl row/col_inds..
    mask = np.logical_and( np.isin(row_ind, sample_users),
                      np.isin(col_ind, sample_movies) )

    sample_sparse_matrix = sparse.csr_matrix((ratings[mask], (row_ind[mask],
col_ind[mask])),
                                             shape=(max(sample_users)+1,
max(sample_movies)+1))

    if verbose:
        print("Sampled Matrix : (users, movies) -- ({} {})".format(len(sample_users),
len(sample_movies)))
        print("Sampled Matrix : Ratings --", format(ratings[mask].shape[0]))

    print('Saving it into disk for furthur usage..')
    # save it into disk
    sparse.save_npz(path, sample_sparse_matrix)
    if verbose:
            print('Done..\n')

    return sample_sparse_matrix
```

4.4.8 XgBoost with Surprise Baseline + Surprise KNNbaseline + MF Techniques

In [0]

```
# prepare train data
x_train = reg_train[['knn_bsl_u', 'knn_bsl_m', 'svd', 'svdpp']]
y_train = reg_train['rating']

# test data
x_test = reg_test_df[['knn_bsl_u', 'knn_bsl_m', 'svd', 'svdpp']]
y_test = reg_test_df['rating']


xgb_all_models = xgb.XGBRegressor(n_jobs=10, random_state=15)
train_results, test_results = run_xgboost(xgb_all_models, x_train, y_train, x_test,
y_test)

# store the results in models_evaluations dictionaries
models_evaluation_train['xgb_all_models'] = train_results
models_evaluation_test['xgb_all_models'] = test_results
```

```
xgb.plot_importance(xgb_all_models)
plt.show()
```

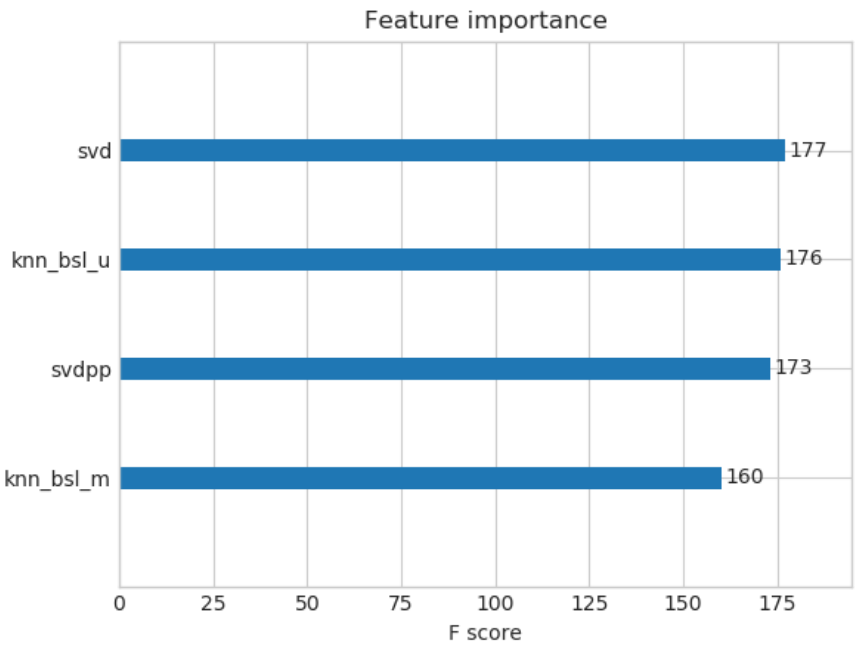
Output (stdout)

```
Training the model..
Done. Time taken : 0:00:01.292225

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.075480663561971
MAPE :  35.01826709436013
```

Text Output

```
<IPython.core.display.Javascript object>
```

HTML Output



Text Output

```
<IPython.core.display.HTML object>
```

4.5 Comparision between all models

In [0]

```
# Saving our TEST_RESULTS into a dataframe so that you don't have to run it again
pd.DataFrame(models_evaluation_test).to_csv('sample/small/small_sample_results.csv')
models = pd.read_csv('sample/small/small_sample_results.csv', index_col=0)
```

```
models.loc['rmse'].sort_values()
```

Text Output

```
svd               1.0726046873826458
knn_bsl_u         1.0726493739667242
knn_bsl_m          1.072758832653683
svdpp             1.0728491944183447
bsl_algo          1.0730330260516174
xgb_knn_bsl_mu    1.0753229281412784
xgb_all_models     1.075480663561971
first_algo        1.0761851474385373
xgb_bsl           1.0763419061709816
xgb_final         1.0763580984894978
xgb_knn_bsl       1.0763602465199797
Name: rmse, dtype: object
```

In [0]

```
print("-"*100)
print("Total time taken to run this entire notebook ( with saved files) is
:",datetime.now()-globalstart)
```

Output (stdout)

```
----------------------------------------------------------------------------------------------------
-------------
Total time taken to run this entire notebook ( with saved files) is : 0:42:08.302761
```

5. FURTHER ASSIGNMENT

In [0]

```
%%javascript
// Converts integer to roman numeral
// https://github.com/kmahelona/ipython_notebook_goodies
// https://kmahelona.github.io/ipython_notebook_goodies/ipython_notebook_toc.js
function romanize(num) {
    var lookup =
{M:1000,CM:900,D:500,CD:400,C:100,XC:90,L:50,XL:40,X:10,IX:9,V:5,IV:4,I:1},
        roman = '',
            i;
        for ( i in lookup ) {
            while ( num >= lookup[i] ) {
                roman += i;
                num -= lookup[i];
            }
        }
        return roman;
 }

// Builds a <ul> Table of Contents from all <headers> in DOM
function createTOC(){
    var toc = "";
    var level = 0;
    var levels = {}
    $('#toc').html('');

    $(":header").each(function(i){
            if (this.id=='tocheading'){return;}
```

```
            var titleText = this.innerHTML;
            var openLevel = this.tagName[1];

            if (levels[openLevel]){
                levels[openLevel] += 1;
            } else{
                levels[openLevel] = 1;
            }

            if (openLevel > level) {
                toc += (new Array(openLevel - level + 1)).join('<ul class="toc">');
            } else if (openLevel < level) {
                toc += (new Array(level - openLevel + 1)).join("</ul>");
                for (i=level;i>openLevel;i--){levels[i]=0;}
            }

            level = parseInt(openLevel);


            if (this.id==''){this.id = this.innerHTML.replace(/ /g,"-")}
            var anchor = this.id;

            toc += '<li><a style="text-decoration:none", href="#' +
encodeURIComponent(anchor) + '">' + titleText + '</a></li>';

        });


    if (level) {
        toc += (new Array(level + 1)).join("</ul>");
    }


    $('#toc').append(toc);

};

// Executes the createToc function
setTimeout(function(){createTOC();},100);

// Rebuild to TOC every minute
setInterval(function(){createTOC();},60000);
```

Text Output

```
<IPython.core.display.Javascript object>
```