КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 124 Системний аналіз

на тему:

УЗАГАЛЬНЕННЯ МЕТОДУ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ З УРАХУВАННЯМ СЕМАНТИЧНОГО ТА ЧАСОВОГО ФАКТОРУ

Виконав студент 2-го курсу магістратури
Шелякін Гліб Вячеславович

(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Івохін Євген Вікторович

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри системного аналізу та теорії прийняття рішень
«04» травня 2023 р.,
протокол № 11
Завідувач кафедри
О. Г. Наконечний

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 65 сторінок, 10 ілюстрацій, 46 джерел посилань.

РЕКОМЕНДАЦІЙНІ СИСТЕМИ, МЕТОД КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ, КЛАСТЕРНИЙ АНАЛІЗ, СІАМСЬКІ НЕЙРОННІ МЕРЕЖІ, МЕТОД СЕМАНТИЧНОЇ ПОДІБНОСТІ, ЧАСОВИЙ ФАКТОР.

Об’єктом дослідження є метод колаборативної фільтрації контенту на основі порівняння об’єктів рекомендацій.

Предметом дослідження є використання часового та семантичного фактору, а також кластеризації даних як засобів впливу на кінцевий результат рекомендацій та покращення якості рекомендованого контенту.

Метою роботи є узагальнити метод колаборативної фільтрації з використанням часового фактору, семантичної подібності та кластеризації даних для покращення якості рекомендацій.

Методи розроблення: наївний метод колаборативної фільтрації; модифікований метод колаборативної фільтрації з урахуванням часового фактору; модифікований метод колаборативної фільтрації з урахуванням семантичної близькості, використовуючи пакет spaCy; метод кластерного аналізу HDBSCAN.

Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки VSCode, мова програмування Python.

Значимість роботи полягає в удосконаленні алгоритму рекомендацій шляхом інтегрування часового та семантичного факторів, а також методу кластерного аналізу з метою зменшення навантаження на систему та покращення якості рекомендацій шляхом відсіювання непотрібного контенту та збереження контексту під час рекомендацій.

Узагальнення алгоритму колаборативної фільтрації з використанням часового та семантичного фактору, а також методу кластерного аналізу, може знайти своє практичне застосування у сервісах інтернет-речей, як от стрімінговий сервіс, інтернет магазин тощо.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Сьогодні існує багато підходів для створення рекомендаційних систем, наприклад, використання нейронних мереж або статистичних математичних моделей. У кожного підходу є свої переваги та недоліки. Так, використання нейронних мереж змушує користувачів сліпо покладатись на вибір контенту, що рекомендується, оскільки нейронні мережі за своєю суттю є “чорними скриньками”, тобто користувачі не знають, кому саме той чи інший контент був рекомендований. З іншого боку, такий підхід не вимагає великої кількості обчислювальних потужностей, як у статистичних моделях.

Щодо статистичних моделей, то вони дають чітку відповідь на питання “Чому саме це було мені рекомендовано?”, що є достатньо великою перевагою як під час розробки системи, так і під час її використання у реальному житті.

Додатковими перевагами статистичних методів створення рекомендаційних систем є їхні здатності до обробки різноманітних форматів даних, включаючи структуровані, напівструктуровані та неструктуровані дані; робота з невеликою кількістю даних, які часто бувають важкими для нейронних мереж; робота з даними, що мають високі рівні шуму, що може негативно впливати на точність рекомендаційних систем, які використовують нейронні мережі; відслідковування змін у поведінці користувачів, оскільки статистичні методи дозволяють адаптувати системи до нових даних та умов; персоналізації рекомендацій на основі конкретних потреб та інтересів користувачів, що може бути складнішим для нейронних мереж.

Серед існуючих методів рекомендацій найяскравішими вважають такі: метод фільтрації на основі вмісту, метод колаборативної фільтрації на основі інформації про користувача (user-based) та на основі порівнянь

об’єктів рекомендацій (item-based).

З урахуванням зростання кількості контенту та користувачів обчислювальні потужності серверів можуть бути недостатніми для ефективної роботи наївного методу. В нашому дослідженні ми пропонуємо враховувати той факт, що інтереси користувачів можуть змінюватись з часом та те, що контент можна заздалегідь розбити на підмножини за певними ознаками, що дозволить системі швидше обробляти дані. Тому в нашій кваліфікаційній роботі ми зробили спробу узагальнення методу колаборативної фільтрації на основі порівняння об’єктів із використанням часового фактору та семантичної подібності, а також методу кластерного аналізу.

Актуальність роботи. За результатами 2022 року IT-індустрія забезпечила валютні надходження до української економіки у $7,34 мільярда. Обсяг експорту збільшився на $400 мільйонів у порівнянні з 2021 роком. В IT Ukraine зазначають, що саме завдяки збереженню темпів зростання, працюючий IT-бізнес підтримує економіку України, створює нові робочі місця, реалізує гуманітарні ініціативи [38].

Розробка рекомендаційних систем є одним з провідних напрямків в галузі IT, оскільки вони сприяють збільшенню прибутків бізнесів завдяки підвищенню обсягу продажів. Рекомендаційні системи є також важливими і для користувачів, оскільки допомагають знайти контент, у якому вони зацікавлені, у величезній кількості наявного матеріалу, використовуючи їх попередній досвід, як от переглянуті фільми, музику, відеоролики або активність в соціальних мережах.

Актуальність дослідження підтверджує і той факт, що у світі існує безліч компаній, які активно використовують рекомендаційні системи, наприклад Spotify, Google, Amazon Music тощо. Крім того, багато веб-сайтів та додатків послуговуються рекомендаційні системи для пропозицій контенту, наприклад, Medium, Goodreads, TikTok тощо.

Важливо також зазначити, що рекомендаційні системи стали предметом багатьох наукових розвідок, вони всебічно досліджуються українськими [5, 7] та іноземними науковцями [8, 9, 10, 12, 17].

Актуальність нашої роботи також полягає в удосконаленні алгоритму рекомендацій шляхом інтегрування часового та семантичного факторів, а також методу кластерного аналізу з метою зменшення навантаження на систему та покращення якості рекомендацій шляхом відсіювання непотрібного контенту та збереження контексту під час рекомендацій.

Мета й завдання роботи. Метою кваліфікаційної роботи є узагальнити метод колаборативної фільтрації з використанням часового фактору, семантичної подібності та кластеризації даних. Для досягнення цієї мети поставлено такі завдання:

1) Проаналізувати основні види колаборативної фільтрації.

2) Узагальнити метод колаборативної фільтрації із застосуванням часового, семантичного факторів та кластеризації даних.

3) Розробити відповідне програмне забезпечення.

4) Перевірити адекватність розробленого методу на даних з різних областей контенту.

5) Проаналізувати отримані результати.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є метод колаборативної фільтрації контенту на основі порівняння об’єктів рекомендацій. При виконанні роботи використовувались такі методи:

1) наївний метод колаборативної фільтрації;

2) модифікований метод колаборативної фільтрації з урахуванням часового фактору;

3) модифікований метод колаборативної фільтрації з урахуванням семантичної близькості, використовуючи пакет spaCy;

4) метод кластерного аналізу HDBSCAN

В якості інструменту створення програмного забезпечення було обрано безкоштовне, вільно поширюване інтегроване середовище розробки VSCode, мова програмування Python.

Можливі сфери застосування. Узагальнення алгоритму колаборативної фільтрації з використанням часового та семантичного фактору, а також методу кластерного аналізу, може знайти своє практичне застосування у сервісах інтернет-речей, як от стрімінговий сервіс, інтернет магазин тощо.

РОЗДІЛ 1 МЕТОДИ, ЩО ЗАСТОСОВУЮТЬСЯ У ДОСЛІДЖЕННІ

1.1 Наївний метод колаборативної фільтрації

Коллаборативна фільтрація - це метод рекомендації контенту, що базується на реакціях користувачів на контент. Вперше метод колаборативної фільтрації був запропонований Голдбергом у 1992 році [20], у вигляді системи фільтрації для електронної пошти Tapestry. Ці реакції можуть бути числовими, як от оцінки товарів в інтернет-магазинах на основі рейтингу, лайки та дизлайки на пости в соціальних мережах, або музику на стрімінгових сервісах; або текстовими, - коментарі користувачів, опис продукту чи товару, тощо.

Головною метою методу є розрахунок оцінки контенту, який користувач ще не бачив, використовуючи інформацію про його попередні реакції. Чим більше оцінка наближається до реальної, тим більш якісним буде кінцевий рекомендаційний висновок. Для того, щоб оцінка системи була більш об'єктивною, необхідно мати якомога більше оцінок користувачів. Таким чином, коллаборативна фільтрація дозволяє зробити більш точні рекомендації користувачам на основі їх власних реакцій на контент.

Із даного визначення випливає, що основними поняттями, що застосовуються у методі є користувач, контент, оцінка та рекомендація.

Користувач – це людина, яка зареєстрована на деякому сервісі та має можливості перегляду, оцінки, покупки контенту на ньому.

Об’єкт – це вміст сайту, з яким користувач може взаємодіяти в рамках цього сервісу та який може бути оцінений користувачами і має метадані, наприклад, опис, назва, вартість, теги, тощо.

Оцінка – це числове значення, що належить множині натуральних чисел, є обмеженим знизу нулем та згори деяким числом з цієї ж множини.

Рекомендація – це множина, що може складатися як з одного, так і з декількох об’єктів, яка буде створена рекомендаційною системою для користувача.

Оцінки користувачів для об’єкту на сервісі легко представити у вигляді матриці R $(rating)$ розмірністю $n \times m$, де m – кількість об’єктів та n – кількість користувачів сервісу.

Нехай існує користувач сервісу u $(user)$, тоді задача полягає у тому, щоб апроксимувати можливу оцінку користувача u для об’єкта i $(item)$.

Процедуру апроксимації можна описати так:

1. Для кожного об’єкта $\hat{i}$ розраховуємо, наскільки він схожий з об’єктом i.
2. Побудуємо множину об’єктів, які найбільш схожі до об’єкту i.
3. Розрахуємо оцінку об’єкта i на основі оцінок з множини найбільш схожих об’єктів.

Тепер розпишемо більш детально кожний з кроків зазначеної процедури. Оскільки кожному об’єкту відповідає стовпчик матриці, то будемо розраховувати міру схожості за стовпцями, використовуючи, наприклад, скориговану схожість косинусів. Формула матиме такий вигляд [2]:

$$sim(i, \hat{i}) = \frac{\sum_{u=1}^{n} (R_{\hat{i},u} - \overline{R}_u) * (R_{i,u} - \overline{R}_u)}{\sqrt{\sum_{u=1}^{n} (R_{\hat{i},u} - \overline{R}_u)^2} * \sqrt{\sum_{u=1}^{n} (R_{i,u} - \overline{R}_u)^2}}$$

Також можна скористатися будь-якою іншою мірою схожості:

- схожість косинусів:

$$sim(i, \hat{i}) = \frac{\sum_{u=1}^{n} R_{i,u} * R_{\hat{i},u}}{\sqrt{\sum_{u=1}^{n} R_{\hat{i},u}} * \sqrt{\sum_{u=1}^{n} R_{i,u}}}$$

- міра Дайса:

$$sim(u, \hat{u}) = \frac{2 * \sum_{u=1}^{n} R_{i,u} * R_{\hat{i},u}}{\sqrt{\sum_{u=1}^{n} R_{\hat{i},u}} + \sqrt{\sum_{u=1}^{n} R_{i,u}}}$$

Отримана величина має такі властивості:

- $sim(i, \hat{i}) \in [0, 1]$ з R;
- $sim(i, \hat{i}) = 0$, якщо користувач не поставив оцінки, або дорівнює 1, якщо оцінки співпали.

Далі оберемо, наприклад, S об'єктів, які найбільш схожі з обраним об'єктом i. Для цього треба відсортувати отримані величини за спаданням та обрати відповідних перших S об'єктів. Сортування нам гарантуватиме те, що перші S об'єктів матимуть величини близькості, які прямують до 1.

Обчислимо, яку можливу оцінку поставив би користувач u об'єкту i. Для цього скористаємось такою формулою:

$$p_{u,i} = \frac{\sum_{\hat{i} \in S} R_{u,\hat{i}} * sim(i, \hat{i})}{\sum_{\hat{i} \in S} sim(i, \hat{i})}$$

Логіка такої оцінки полягає у тому, що чим міра схожості $sim(i, \hat{i})$ ближча до об'єкту i, тим більший вклад об'єкту $\hat{i}$ до остаточної оцінки.

Проаналізувавши наївний метод колаборативної фільтрації, стали очевидними його переваги та недоліки.

1. Проблема "холодного старту", яка виникає у тому випадку, коли контенту не була присвоєна оцінка жодним користувачем, через що система просто не звертає на нього увагу до того моменту, поки хоча б один користувач не оцінить цей контент. Вона є досить відомою і досліджуваною [13, 14, 27, 29].

Виділяють 3 категорії “холодного старту”:

1. Нова спільнота: коли запускається нова система, у каталозі може бути багато елементів, але мало взаємодії та присутності користувачів, що ускладнює надання надійних рекомендацій.

2. Нові елементи: нові елементи системи, може бути відповідна інформація про контент, але немає взаємодії з користувачем.

3. Нові користувачі: нові користувачі можуть входити в систему без будь-якої взаємодії з системами та без будь-яких персоналізованих рекомендацій для гостей.

Для вирішення проблем холодного старту пропонують використовувати різні методи, наприклад, трансферне навчання [34], активне навчання [14], гібридизація рекомендаційної системи у поєднанні із контентної та колаборативної фільтрації [16], використання контексту, в якому створюються та надаються рекомендації (демографічні дані, час та дата, тощо) [25]. У наступних розділах ми запропонуємо можливі способи розв’язання цієї проблеми.

2. Проблема масштабування системи, оскільки у випадку великої кількості як користувачів так і об’єктів (наприклад, 1 млн. користувачів та 1 млн. об’єктів), задача стає транс обчислювальною та потребує дуже великої кількості ресурсів і часу на обрахунки, що унеможливлює швидкі рекомендації. Частковий розв’язок цієї задачі буде розглянутий у наступних розділах.

3. Так звана “синонімія” об’єктів, які за своєю суттю є однаковими, але мають, наприклад, різні теги, назви, описи, тощо.

Тепер перейдемо до розгляду переваг методу колаборативної фільтрації.

1. Незалежність від конкретної доменної області. Реалізація методу не залежить від специфіки його використання, що дозволяє застосовувати його в будь-якій сфері, майже не змінюючи математичний апарат.

2. Прозорість роботи методу. Відмінності від моделей машинного навчання дають змогу перевірити на кожній ітерації системи правильність

обрахунків та логіки методу, що значно спрощує розробку програмного забезпечення системи рекомендацій.

3. Не потребує великої кількості вхідних параметрів. Для коректної роботи методу достатньо матриці зворотних зв'язків (оцінок користувачів), що відрізняє його від систем, які потребують значно більшої кількості вхідних параметрів.

1.2 Метод семантичної подібності

У попередньому розділі ми розглянули процедури знаходження коефіцієнта схожості об’єктів, використовуючи явні оцінки користувачів, які можуть мати різні значення, незважаючи на те, що об’єкти є однаковими. Втім не слід забувати про такі атрибути, як, наприклад, опис, назва тощо, які також відіграють важливу роль у рекомендаціях, оскільки допомагають визначити, якою мірою один об’єкт схожий із іншими.

Тому у цьому розділі ми розглянемо один із методів, що використовується для перетворення текстових оцінок-характеристик у числові значення за допомогою машинного навчання, яке є потужним інструментом для обробки природної мови (NLP), оскільки воно може аналізувати та використовувати статистичні властивості тексту для розв'язання складних завдань, таких як машинний переклад, кластеризація текстів, сентимент-аналіз та багато інших.

Зауважимо на основних перевагах та недоліках методів машинного навчання. Основною причиною такого вибору є те, що машинне навчання у порівнянні зі стемінгом Портера та іншими традиційними методами обробки тексту, які засновані на правилах та евристиках, що можуть не враховувати складні взаємозв'язки між словами та контекстом, не працювати з даними, які не відповідають стандартним граматичним правилам, що знижує їх ефективність, методи машинного навчання мають кілька переваг [21]:

1. Автоматизація процесу аналізу тексту. Машинне навчання може автоматично аналізувати велику кількість тексту і витягувати значущі ознаки, які можуть бути використані для класифікації, кластеризації, та інших завдань NLP.

2. Висока точність результатів. Машинне навчання може забезпечити високу точність при аналізі тексту, оскільки воно базується на статистичних алгоритмах, які здатні працювати з великою кількістю даних та визначати складні залежності.

3. Підтримка мовної різноманітності. Машинне навчання може працювати з різними мовами та враховувати відмінності між ними, що дозволяє створювати універсальні моделі, які можуть застосовуватися в різних контекстах.

4. Додатковою перевагою методів машинного навчання є також можливість використовувати глибинне навчання (deep learning) для обробки тексту. Глибинне навчання полягає у створенні нейромереж, які можуть вчитися з даних та визначати складні залежності між ними. Застосування глибинного навчання в NLP дозволяє створювати більш точні та складні моделі, які здатні вирішувати завдання, що раніше були неможливими для традиційних методів обробки тексту.

5. Методи машинного навчання дозволяють використовувати векторну репрезентацію слів, яка дозволяє враховувати не тільки лексичну схожість між словами, але й семантичну схожість. Наприклад, слова "автомобіль" та "машинка" можуть мати більш близьку векторну репрезентацію, ніж "автомобіль" та "книга". Це дозволяє створювати більш точні та ефективні моделі для NLP.

6. Можливість покращуватися з часом. Якщо модель навчається на нових даних, вона може покращувати свою точність та здатність розуміти мову. Це дозволяє створювати моделі, які можуть адаптуватися до змінних умов та навчатися на нових даних.

Проте, методи машинного навчання мають і свої недоліки. Наприклад, вони можуть бути вимогливі до ресурсів комп'ютера та часу для навчання. Крім того, вони можуть бути більш складні для розуміння та налаштування, що може потребувати спеціалістів з машинного навчання.

В нашій роботі ми досліджуємо метод семантичної подібності, який базується на вимірюванні подібності двох текстів на основі значення слів, що в них використовуються. Метод семантичної подібності використовує Word Embeddings, який представляють кожне слово як вектор у багатовимірному просторі, фіксуючи значення та контекст слів. Семантичну подібність вимірюють за допомогою таких алгоритмів, як відстань перенесення слова (WMD), сіамські нейронні мережі (SNN) або відстань Жаккара [28].

Розглянемо більш детально принцип роботи кожного із зазначених алгоритмів. алгоритму “Відстань перенесення слова” (Word Mover's Distance) — це міра семантичної подібності між двома текстами, яка базується на концепції «відстані землероба» з обчислювальної геометрії [32]. WMD використовується для визначення мінімальної «вартості» перетворення слів одного тексту в слова іншого тексту, де ціною є «відстань» між двома словами у просторі вбудовування.

Щоб обчислити WMD між двома текстами, ми спочатку представляємо кожне слово в текстах як вектор у просторі вбудовування, наприклад Word2Vec або GloVe. Потім ми обчислюємо «відстань» між векторами кожного слова в двох текстах за допомогою метрики відстані, такої як евклідова або косинусна відстань [37].

По-перше необхідно розрахувати дискретний імовірнісний розподіл слів шляхом нормалізації вектор сумки слів:

$$n_{L1}(x) = x / \sum_i x_i$$

Алгоритм визначає матрицю вартості такого вигляду:

$$C \in R^m \times R^m$$

Як відстань між вкладеннями, наприклад:

$$C_{ij} = ||z_i - z_j||_2$$

Відстань між двома документами x та x' є оптимальним рішенням наступної задачі:

$$\frac{minimize}{P \in R^{m \times m}} \sum_{ij} C_{ij} P_{ij}$$

$$P_{ij} \geq 0,\ P1 = n_{L1}(x),\ P^T 1 = n_{L1}(x')$$

де P_{ij} кількість слів i, що переносяться до слова j.

Тоді WMD розраховується як мінімальна «вартість» переміщення слів одного тексту до слів іншого тексту, де ціна — це відстань між векторами слів. Ця вартість розраховується за допомогою угорського алгоритму, який знаходить оптимальний збіг між словами в двох текстах.

Метрика Жаккара , також відома як індекс Жаккара або коефіцієнт подібності Жаккара, є показником подібності, який використовується для вимірювання подібності чи відмінності між двома наборами величин. Він визначається як розмір перетину двох множин, поділений на розмір об’єднання множин [37].

У математиці коефіцієнт подібності Жаккара між двома наборами A і B дорівнює: $J(A, B) = |A \cap B| / |A \cup B|$, де |A| та |B| позначають потужності (кількість елементів) множин A і B відповідно, а $|A \cap B| / |A \cup B|$ позначимо потужності перетину та об'єднання множин A та B відповідно.

Коефіцієнт подібності Жаккара може приймати значення від 0 до 1, де 0 означає, що два набори не мають спільних елементів, а 1 означає, що

два набори ідентичні. Чим вищий коефіцієнт подібності Жаккара, тим більш схожими вважаються два набори.

Було встановлено, що WMD перевершує традиційні заходи семантичної подібності, як от відстань Жаккара, у таких програмах, як от виявлення плагіату або кластеризація документів і розширення запитів. Однак розрахунок WMD може бути дорогим з обчислювальної точки зору, особливо для довгих текстів або великих словників.

Наступний метод, який ми будемо використовувати у дослідженні - сіамські нейронні мережі, які давно відомі та широко використовуються у задачах для порівнянь двох наборів даних [15, 36] Вони є класом нейронних мереж, що використовуються для порівняння двох вхідних даних і визначення їх схожості. Ці мережі називаються сіамськими, тому що вони складаються з двох ідентичних підмереж, кожна з яких обробляє один із входів, а потім об'єднує свої виходи.

Розглянемо більш детально структуру та принцип роботи сіамських нейронних мереж. На вхід при тренуванні мережа приймає три значення (w_1, w_2, s), де w_1 та w_2 – векторні представлення тексту, $s \in \{0,1\}$ – очікуваний показник схожості між w_1 та w_2 (1 – схожі, 0 – ні). Кожній підмережі передається один вектор, обидва вектори проходять одну і ту саму обробку (кількість шарів у мережах, їх вид та параметри повністю співпадають). Останній, спільний шар, отримує на вхід дві результуючі ознаки та визначає їх схожість, вимірюючи Манхеттенську відстань між двома ознаками [39, 40]: $exp(-|v_1 - v_2|) \in [0, 1]$, де v_1 та v_2 -– ознаки, що отримуються на виході з підмереж. Метою тренування є якомога менша відстань для схожих та якомога більша для несхожих об'єктів.

Сіамська мережева архітектура часто використовується в таких програмах, як подібність тексту, схожість зображень, перевірка підпису та виявлення шахрайства. У випадку подібності тексту мережа приймає два

тексти як вхідні дані, кодує їх за допомогою спільного рівня вбудовування, а потім порівнює закодовані вектори, щоб визначити їх подібність.

Одна з ключових переваг сіамських мереж полягає в тому, що їх можна навчити на парах вхідних даних із відомою схожістю, що робить їх придатними для вивчення показників подібності з позначених даних. Процес навчання передбачає мінімізацію контрастної функції втрат, яка заохочує мережу навчитися розрізняти подібні та несхожі пари вхідних даних.

Було показано, що сіамські мережі добре працюють у різноманітних завданнях, які вимагають вимірювання подібності, таких як зіставлення тексту та пошук зображень. Вони також були розширені до більш складних архітектур, таких як триплетні мережі, де використовуються три входи замість двох, і мережа вчиться порівнювати відстань між позитивним і негативним прикладом.

1.3 Метод кластерного аналізу HDBSCAN

Наразі існує багато методів кластеризації, такі як: OPTICS, DBSCAN, HDBSCAN, Agglomerative, K-means, K-means++ та інші. Всі вони знайшли використання у різних галузях програмування. Втім, існуючі методи мають ряд обмежень:

1. Деякі методи (наприклад, DBSCAN [18, 22]) можуть забезпечити лише «пласке» (тобто неієрархічне) маркування об’єктів даних на основі глобального порогу щільності. Використання одного порогу щільності часто не може належним чином охарактеризувати загальні набори даних кластери дуже різної щільності та/або вкладені кластери.

2. Серед методів, які забезпечують ієрархію кластеризації, деякі (наприклад, gSkeletonClu [35]) не здатні автоматично спростити ієрархію до представлення, яке легко інтерпретувати, залучаючи лише найважливіші кластери.

3. Багато ієрархічних методів, включаючи OPTICS [43] і gSkeletonClu, пропонують лише те, як виділити плоский розділ за допомогою глобального порогу розрізу/щільності, що може не призвести до найбільш значущих кластерів, якщо ці кластери характеризуються різними рівнями щільності.

4. Деякі методи обмежені конкретними класами проблем, такими як мережі (gSkeletonClu) і набори точок у реальному просторі координат (наприклад, DECODE [30] і Generalized Single-Linkage [31, 33]).

5. Більшість методів залежить від кількох, часто критичних вхідних параметрів.

Тому ми обрали метод кластеризації на основі щільності, який є популярною парадигмою кластеризації [36] завдяки значним перевагам.

По-перше, HDBSCAN має вбудований механізм підбору параметрів, що дозволяє знайти оптимальні значення параметрів, такі як мінімальна густота і радіус, автоматично, що робить його простим у використанні та зменшує необхідність у ручному налаштуванні параметрів алгоритму.

По-друге, у HDBSCAN є також додаткові переваги, зокрема він швидкий і масштабований до великих наборів даних, що робить його ідеальним для великих датасетів. Крім того, HDBSCAN є відкритим джерелом і доступний для використання безкоштовно.

По-третє, HDBSCAN може виявляти кластери будь-якої форми, включаючи складні, що переплітаються або мають вузькі проміжки.

Четверте, HDBSCAN здатен розрізняти шумові точки від дійсних кластерів, що дозволяє отримати більш точні результати.

Розглянемо роботу алгоритму.

1. Побудова "матриці сусідства"- обчислення відстані між кожною парою точок і створює матрицю сусідства. Це дозволяє відокремити більш щільні області від менш щільних областей.

2. Побудова "дерева мінімальних спанів" - обчислення дерева мінімального охоплення на основі матриці сусідства. Дерево мінімального охоплення - це граф, який з'єднує всі точки, але має найменшу можливу загальну вагу.

3. Побудова "дерева сплитів" - обчислення дерева сплитів з дерева мінімальних спанів. Дерево сплитів розділяє кластери на підкластери і збільшує точність кластеризації.

4. Вибір кластерів - на основі дерева сплитів обчислення набору кластерів, які найкраще описують дані.

Алгоритм кластеризації HDBSCAN є потужним інструментом для кластеризації даних, який вирішує проблеми, пов'язані з визначенням кількості кластерів, шумом і формою кластерів. Це підхід, який використовує ієрархічну модель густини, щоб знаходити кластери, які можуть мати будь-яку форму.

Таким чином у даному розділі нами розглянуто та проаналізовано методи, алгоритми та підходи до розв'язання поставлених задач, в результаті чого були обрані: алгоритм кластеризації HDBSCAN для проведення кластерного аналізу; метод колаборативної фільтрації на основі вмісту як об'єкт дослідження; метод сіамських нейронних мереж для розрахунку семантичної подібності.

РОЗДІЛ 2. УЗАГАЛЬНЕННЯ МЕТОДУ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ З УРАХУВАННЯМ ЧАСОВОГО ФАКТОРУ ТА СЕМАНТИЧНОЇ ПОДІБНОСТІ

2.1 Постанова задачі для узагальненого метода

Нехай об’єктами рекомендацій будуть деякі сутності, характеристики яких можна розділити на три групи:

1. Оцінки. Під оцінками ми розуміємо вектор відповідностей між деякими користувачами та деяким об’єктом. Ця відповідність виражається у вигляді деякого значення. Передбачається, що відповідне значення є обмеженим знизу та згори. Наприклад від 1 до 5, від 0 до 1 тощо.

2. Текстова інформація. Під текстовою інформацією ми розуміємо деякий скінченний список характеристик деякого об’єкту, що представлена у вигляді тексту. Наприклад, це можуть бути описи, назви, резюме тощо.

3. Часова інформація. Під часовою інформацією ми розуміємо дату та час виконання певної дії деякого користувача над деяким об’єктом. Наприклад, це може бути оцінка об’єкту, перегляд об’єкту, рецензування об’єкту, відгук до об’єкту, замовлення об’єкту тощо.

Тоді наша задача полягає у тому, щоб розробити такий алгоритм, який би міг без обмежень на доменну область робити апроксимації оцінок користувачів з використанням вищезазначених вимог, та робити це швидше, ніж стандартний метод колаборативної фільтрації.

2.2. Узагальнення часового фактору

Для того, щоб врахувати часовий фактор, ми будемо використовувати функцію “старіння” вигляду $f(t) = e^{-t}$, $0 \leq t < +\infty$, $f(t) \in [0; 1]$. Ця функція гарантує те, що чим більше значення t, тим менше значення буде на виході, оскільки $\lim\limits_{t \to \infty} f(t) = 0$. Розглянемо алгоритм використання функції старіння:

1. Розрахувати для кожного об’єкта значення функції “старіння” для показника часової інформації.

2. Відсортувати отримані значення за спаданням. Це нам гарантуватиме, що об’єкти, які були переглянуті нещодавно, матимуть більший вплив на рекомендацію.

3. Далі оберемно перші *V (viewed*) об’єктів, які наразі знаходяться у тренді користувача та відповідно мають більший вплив на те, що він хоче бачити.

Важливо додати, що при розрахунку функції старіння ми будемо використовувати не точний час до секунд, хвилин, годин, а до днів. Це дозволить нам краще утримувати контекст на тенденцію користувача.

Це не лише пришвидшить обчислення, а й зменшить навантаження на систему, наприклад, сервер чи компьютер.

Процедура обчислення оцінки, використовуючи часовий фактор, матиме такий вигляд:

1. Для кожного об’єкта $\hat{i} \in V$ розраховуємо, наскільки він схожий з об’єктом i.
2. Побудуємо множину об’єктів, які найбільш схожі з об’єктом i.
3. Розрахуємо оцінку об’єкта i на основі оцінок з множини найбільш схожих об’єктів та часового вагового коефіцієнту.

Перші два кроки розраховуються аналогічно до стандартного методу колаборативної фільтрації, а ось розрахунок остаточної оцінки буде таким:

$$p_{u,i} = \frac{\sum_{\hat{i} \in V} R_{u,\hat{i}} * sim(i,\hat{i}) * f(t_{\hat{i}})}{\sum_{\hat{i} \in V} sim(i,\hat{i}) * f(t_{\hat{i}})}$$

При цьому треба зазначити, що введення додаткового множника не змінює початкових властивостей функції, що дозволяє легко використовувати дану оцінку у подальшій розробці.

Ця модифікація наївного методу колаборативної фільтрації дозволяє уникнути проблеми масштабування системи, оскільки ми обробляємо не всю множину об’єктів, а лише останні V переглянутих об’єктів, що дає можливість пришвидшити роботу системи в декілька разів в залежності від кількість оцінок та об’єктів.

Також використання функції “старіння” дає системі можливість бути більш адаптивною до змін смаків користувачів сервісу, приділяючи більше уваги тому контенту, який був переглянутий нещодавно, та менше уваги тому, що був переглянутий колись, наприклад, тиждень, місяць, рік потому тощо. Таким чином, часова інформація може допомогти рекомендаційній системі уникнути пропонування застарілого контенту користувачам.

Ще часовий фактор дозволить покращити якість рекомендацій, оскільки система буде спиратися на тренди користувача, які він задає, під час перегляду контенту різної тематики, що значно вплине на різноманіття контенту, що рекомендується. До таких трендів, наприклад, можна віднести новизну, коли користувачі більше зацікавлені в новому контенті. В цьому випадку врахування часової інформації допомагає рекомендаційній системі пропонувати свіжі матеріали користувачам.

Наступним трендом можна назвати сезонність, коли деякі типи контенту можуть бути більш популярні в певний час року. Наприклад, фільми про Різдво можуть бути більш популярні під час святкувань Різдва, тому рекомендаційна система може врахувати дату, щоб запропонувати такий контент у відповідний період часу.

2.3 Узагальнення використання методу семантичної подібності

В розділі 1.2 ми описали метод семантичної подібності (сіамські нейронні мережі), який використовують для того, щоб розраховувати коефіцієнт подібності між двома текстовими даними. Опишемо, яким чином його можна застосувати для апроксимації остаточної оцінки користувача.

Нехай M – множина текстової інформації, що відображає суть деякого об’єкту. Кожен елемент множини є пара $(a_1, a_2, ..., a_k)$ – де a_i – це деяке текстове представлення i – го атрибуту поточного об’єкту. Це можуть бути описи, рецензії, набір технічних характеристик, відгуки.

Тоді міру близькості двох об’єктів можна представити такою формулою:

$$sim_{meta}(i, \hat{i}) = \frac{\sum_{j=1}^{k} sim(i_j, \hat{i}_j)}{n}$$

Де $sim(i_j, \hat{i}_j)$ – коефіцієнт подібності j – х характеристик об’єктів i та $\hat{i}$ із застосуванням сіамських нейронних мереж у якості оцінювачів.

Введемо матрицю $OS\ (Object\ Similarity) \in R^{n,n}$, де n – кількість об’єктів на сервісі. Матриця є квадратною та симетричною. $OS_{i,j}\ i > j = \overline{1, n-1}$ – оцінки схожості i – го об’єкту до j – го об’єкту. Це дає нам змогу зберігати результати у базі даних, а не в оперативній пам'яті, що гарно впливає на швидкодію, а також розраховувати оцінку в офлайн режимі, оскільки кількість об’єктів зростає доволі повільно, що не буде створювати проблеми в оновлені даних. Це в свою чергу надає нам такий список переваг, у порівнянні з тим, щоб зберігати дані в оперативній пам'яті [48]:

1. Масштабованість. Бази даних можуть обробляти великі обсяги даних і запитів, тому вони є більш масштабованими, ніж оперативна пам'ять. Фізичні габарити обладнання обмежують розмір оперативної пам'яті, тоді як бази даних можуть бути масштабовані велику кількість комп'ютерів чи серверів, що дозволяє обробляти великі обсяги даних та запитів, які неможливо обробити з використанням оперативної пам'яті.

2. Безпека. Бази даних забезпечують захист даних від випадкового або навмисного знищення, наприклад, через аварійне вимкнення живлення. У базі даних можна використовувати різні методи захисту даних, такі як резервне копіювання даних, шифрування, контроль доступу, автентифікація, що дозволяє зберігати дані в безпеці.

3. Консистентність. Бази даних можуть забезпечити транзакційну консистентність даних. Це означає, що база даних забезпечує виконання всіх транзакцій з виконанням установлених правил. Якщо якась транзакція не відповідає встановленим правилам, вона не буде виконана, що забезпечує коректність даних. Крім того, бази даних дозволяють використовувати консистентність даних для забезпечення їх цілісності та точності.

4. Доступність. Бази даних можуть забезпечити одночасний доступ до даних багатьма користувачами та додатками.

Тепер можемо сформулювати процедуру підрахунку оцінки, використовуючи семантичну подібність:

1. Для кожного об’єкта $\hat{i}$ розрахуємо наскільки він схожий з об’єктом i та занесемо отримані значення у матрицю OS.
2. Побудуємо множину об’єктів, які найбільш схожі з об’єктом i з множини останніх переглянутих об’єктів V.
3. Розрахуємо оцінку об’єкта i на основі оцінок з множини найбільш схожих об’єктів.

Тоді остаточна оцінка матиме такий вигляд:

$$p_{u,i} = \frac{\sum_{\hat{i} \in V} R_{u,\hat{i}} * sim_{meta}(i, \hat{i})}{\sum_{\hat{i} \in V} sim_{meta}(i, \hat{i})}$$

Ця модифікація колаборативної фільтрації наївного методу розв'язує проблему "холодного старту", що позитивно впливає на якість

рекомендацій користувачам у випадку, коли об'єкт ще не був оцінений жодним користувачем після його додавання до сервісу. Крім того, вона уникає проблеми “синонімії” об'єктів в рекомендаціях, що також покращує остаточний вибір рекомендацій.

2.4 Узагальнення використання методу кластерного аналізу

Використання алгоритму кластеризації для методу колаборативної фільтрації обумовлено тим, що за припущенням, користувачі, які однаково оцінювали будь-які об’єкти в минулому, мають схильність давати схожі оцінки і у майбутньому. Для цього ми будемо використовувати зазначену у попередньому розділі матрицю OS та алгоритм кластеризації HDBSCAN.

Це пояснюється тим, що хоча об’єкти можуть належати до різних груп за певними характеристиками, наприклад, жанри, категорії тощо, за текстовою інформацією вони можуть належати до однієї спільної групи.

Текстова інформація може бути більш детальною та багатоаспектною, що дозволяє краще розуміти, що саме сподобалося користувачам та забезпечити більш адаптивний та персоналізований досвід, аніж використання інших характеристик.

Крім того, текстова інформація може бути використана для знаходження спільних тем або ключових слів між різними об'єктами, що допомагає виявляти зв'язки та схожості між ними. Наприклад, якщо користувачі високо оцінюють фільми, які містять ключові слова "наукова фантастика" або "пригоди", то система може рекомендувати інші фільми з подібними характеристиками. Або, якщо ми розглядаємо товари в інтернет-магазині, то з текстовим описом товарів ми можемо більш детально охарактеризувати їх властивості та функції. Ще одним прикладом може бути аналіз поведінки користувачів в соціальній мережі. У цьому випадку текстові повідомлення можуть допомогти нам зрозуміти, які теми цікавлять користувачів, які групи користувачів пов'язані між собою, які інтереси вони мають та інше.

Але на відміну від текстової інформації інші метрики, такі як рейтинг, час перегляду або кількість переглядів, можуть бути корисними, але не дадуть повної картини про те, що саме сподобалося користувачам у даному об'єкті. Наприклад, користувач може не дивитись фільм до кінця, але залишити позитивний відгук про нього, що свідчить про те, що йому сподобався сюжет або якість акторської гри. Або, якщо користувач поставив погану оцінку товару через проблеми доставки сервісу, а не через те, що товар сам по собі є поганим. Ще одним прикладом може бути те, що кількість вподобайок на постах у соціальних мережах не завжди відображає дійсність, оскільки вподобайки можуть бути штучними, тобто такими, що не виставлені реальними користувачами сервісу.

Таким чином, використання текстової інформації як метрики дає змогу зробити більш точні та персоналізовані рекомендації, що підвищує якість обслуговування та задоволення користувачів.

Вибір алгоритму HDBSCAN також обумовлений і тим, що він заснований на ієрархічній побудові кластерів з урахуванням щільності об’єктів у багатовимірному просторі. Також важливо зазначити, що цей алгоритм потребує на вхід лише один параметр, який впливає на його роботу – найменша кількість об’єктів, яка може вважатися кластером.

Сформулюємо процедуру попередньої обробки даних для методу колаборативної фільтрації на основі порівнянь об’єктів:

1. Побудуємо матрицю подібності об’єктів з використанням наявних даних у нашій вибірці.
2. Використовуючи алгоритм кластеризації, розіб’ємо множину об’єктів на відповідні кластери.
3. Збережемо відповідні кластери для використання у майбутньому.

Описана вище процедура називається агрегацією даних. Агрегація даних перед початком роботи алгоритму може зменшити час виконання та

складність обчислень. Коли алгоритм працює з необробленими даними, він повинен виконувати додаткову роботу з обробки та групуванням даних під час виконання. Це може зайняти багато часу та ресурсів обчислювального простору.

Крім того, якщо дані агрегуються перед початком роботи алгоритму, можна виконати оптимізацію та валідацію даних, що може допомогти уникнути помилок та поганих результатів. Наприклад, можна видаляти дублікати, перевіряти на наявність відсутніх значень, проводити перетворення даних та інші дії для підготовки даних до виконання алгоритму.

Додатково, агрегація даних перед початком роботи алгоритму може забезпечити зручність використання та обміну даними між різними додатками та системами. Агреговані дані можна зберігати в базі даних або файловій системі, що може бути доступним багатьом користувачам та додаткам одночасно.

Отже, це дозволяє нам, по-перше, зменшити час виконання рекомендацій, оскільки множина об’єктів у кластері набагато менша, ніж їх загальна кількість, по-друге, при додаванні нового об’єкта ми швидше можемо знаходити, якому кластеру він належить.

Також треба зазначити, що запропонована модифікація дозволяє нам уникнути проблеми масштабованості, оскільки нам не треба буде обробляти всю матрицю зворотних зв’язків, а лише її частину, що суттєво пришвидшить роботу методу, особливо на великій кількості об’єктів.

2.5 Опис отриманого алгоритму

Дослідивши методи оцінювання подібності у попередніх розділах, ми можемо вивести остаточний алгоритм підрахунку оцінки об’єкту.

Нехай $u(i, \hat{i}) = \alpha * sim_{mark}(i, \hat{i})$, де $sim_{mark}(i, \hat{i}) \in [0, 1]$ з R – міра схожості об’єктів за оцінками користувачі, а $\alpha \in [0, 1]$ з R –

ваговий коефіцієнт, який відповідає за те, наскільки величина $sim_{mark}(i, \hat{i})$ відіграватиме роль в остаточній оцінці.

Нехай $v(i, \hat{i}) = \beta * sim_{meta}(i, \hat{i})$, де $sim_{meta}(i, \hat{i}) \in [0, 1]$ з R – міра схожості об'єктів, заснована на порівнянні їх метаданих, а $\beta \in [0, 1]$ з R – ваговий коефіцієнт, який відповідає за те, наскільки величина $sim_{meta}(i, \hat{i})$ відіграватиме роль в остаточній оцінці.

Тоді ми можемо записати остаточну оцінку (гібридну), використовуючи вищезгадані величини:

$$sim_{hybrid} = \frac{(u+v)}{2}$$

Перевіримо, чи зберігаються властивості запропонованої оцінки подібності відносно оцінки методу колаборативної фільтрації на основі порівнянь об'єктів:

1. $u = 0, v = 1 \Rightarrow sim_{hybrid} \in [0, 1]$
2. $u = 1, v = 0 \Rightarrow sim_{hybrid} \in [0, 1]$
3. $u = 0, v = 0 \Rightarrow sim_{hybrid} \in [0, 1]$
4. $u = 1, v = 1 \Rightarrow sim_{hybrid} \in [0, 1]$

Отже, можна зробити висновок, що запропонована оцінка подібності повністю співпадає з властивостями оцінки методу колаборативної фільтрації на основі порівнянь об'єктів.

Побудуємо остаточну формулу розрахунку оцінки користувача:

$$p_{u,i} = \frac{\sum_{\hat{i} \in V} sim_{hybrid}(i, \hat{i}) * f(t_{\hat{i}}) * R_{u,\hat{i}}}{\sum_{\hat{i} \in V} sim_{hybrid}(i, \hat{i}) * f(t_{\hat{i}})}$$

Також побудуємо остаточну процедуру роботи запропонованого алгоритму як для заздалегідь визначеної множини об’єктів, так і для новододаних об’єктів.

Процедура для заздалегідь визначеної множини об’єктів:

1. Застосувавши алгоритм кластеризації, розіб’ємо множину об’єктів на відповідні кластери.
2. Оберемо об’єкт, для якого будемо робити рекомендацію.
3. Оберемо всі об’єкти з відповідного кластеру, окрім того, для якого робиться рекомендація.
4. Побудуємо множину оцінок, яку користувач міг би поставити фільмам з множини зазначеної вище.
5. Відсортуємо відповідну множину оцінок за спаданням.
6. Оберемо K найкращих оцінок та об’єктів відповідно до цих оцінок.

Процедура для новододаних об’єктів:

1. За текстовою інформацією про об’єкт додамо його до одного із заздалегідь визначеного кластеру.
2. Оберемо всі об’єкти з відповідного кластеру, окрім того, для якого робиться рекомендація.
3. Побудуємо множину оцінок, яку користувач міг би поставити фільмам з множини зазначеної вище.
4. Відсортуємо відповідну множину оцінок за спаданням.
5. Оберемо K найкращих оцінок та об’єктів відповідно до цих оцінок.

Таким чином у цьому розділі нами зроблена спроба розробити алгоритм, який би міг без обмежень на доменну область робити апроксимації оцінок користувачів швидше та якісніше, ніж стандартний метод колаборативної фільтрації, завдяки введенню обмежень на типи даних; зміні формули та методу розрахунку семантичної подібності;

часовому фактору для збереження тенденцій користувача; попередній обробці даних у вигляді застосування методу кластеризації на основі щільності та ієрархії для розбиття множини об’єктів на відповідні споріднені підгрупи.

Програмна реалізація алгоритму та його тестування у порівнянні зі стандартним методом колаборативної фільтрації будуть розглянуті у наступному розділі.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ УЗАГАЛЬНЕНОГО МЕТОДУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Вибір мови програмування та пакетів для програмної реалізації

Для вирішення поставленої задачі нами було обрано мову програмування Python. Можливості Python досліджувались багатьма українськими та іноземними фахівцями в галузі IT [1, 3, 4, 23, 26]. У всіх наукових розвідках відмічається, що мова Python є простою у використанні, але в той же час має багато інструментів для структурування та підтримки. Вона має перевагу над багатьма іншими мовами програмування, оскільки ефективніше обробляє помилки та є дуже високорівневою, має вбудовані типи даних високого рівня, такі як масиви та словники з динамічними розмірами. Реалізація цих типів даних на мовах, таких як от C++, вимагає значних зусиль, часу, та для ефективної роботи програми – неабиякий досвід у розробці програмного забезпечення. Python також має велику бібліотеку стандартних модулів, які можуть слугувати основою для нових програм або прикладами під час вивчення мови.

Використання інтерпретатора мови дозволяє взаємодіяти з мовою програмування у режимі реального часу. Це надає можливість експериментувати з можливостями мови, створювати шаблони програм або тестувати функції при розробці. Зазвичай, програми, написані мовою Python, є значно коротшими в порівнянні з еквівалентами на C++, зокрема з таких причин:

1. Вбудовані типи даних високого рівня дозволяють виконувати складні операції однією інструкцією;

2. Групування інструкцій виконується за допомогою відступів замість фігурних дужок;

3. Змінним, що оголошуються, немає потреби строго прописувати їх типізацію, що в свою чергу дозволяє не витрачати багато часу на написання програмного забезпечення.

Python – мова програмування, яка підтримує розширення. Завдяки цьому, можна використовувати знання C++ для створення нових функцій та модулів, які потім можна вбудовувати в Python. Більше того, інтерпретатор Python може використовуватися як командна мова або розширення для програм, написаних на C++. Це дає можливість програмістам використовувати Python разом з C++, отримуючи переваги обох мов такі як:

1. Швидкодія, завдяки низкорівневій архітектурі C++

2. Зручність та легкість використання, завдяки високорівневій архітектурі Python

Зараз мовою Python користується значна кількість програмістів по всьому світу, і це число зростає щороку. Згідно з Analytics Insight, яка у 2023 році провела дослідження про використання існуючих мов програмування у галузі IT індустрії, Python займає друге місце після JavaScript з кількістю розробників дев'ять мільйонів [39]. Це обумовлено зокрема і тим, що Python приваблює користувачів своєю універсальністю – вона працює на Windows, UNIX та Macintosh і може бути використана для розробки як невеликих, так і великих програм.

У результаті ми можемо зробити висновок, що мова програмування Python є достатньо простою у використанні, має динамічну типізацію та доброзичливий інтерфейс для користувача, а також надає велику кількість засобів для структурування та підтримки великих програм, які можуть бути застосовані для широкого спектру задач.

Для програмної реалізації алгоритму кластерного аналізу нами було обрано пакет python-hdbscan.

Ця програмна реалізація, на відміну від інших пакетів, що реалізують цей алгоритм, відрізняється високою швидкістю виконання операцій, зручним інтерфейсом взаємодії, та великою кількістю додаткових функцій як от:

1. Генерація штучних даних для тестування алгоритму на основі параметрів щільності точок навколо деякої центроїди, яку користувач може задати самостійно.

2. Пошук викидів з заданої користувачем множини, що дозволяє використовувати цей алгоритм не лише як алгоритм кластеризації, а і як алгоритм для пошуку викидів.

3. Велика кількість наявних метрик, завдяки яким розраховуються матриці відстаней таких як: L1, L2, Манхетенська метрика, метрика Мінковського,відстань Хеммінга [40].

На наш погляд, основною перевагою з точки зору виконання поставленої задачі є те, що цей пакет дозволяє у зручному вигляді зберігати відповідні побудовані кластери, а саме у JSON форматі. Це дозволяє легко та швидко обробляти отримані результати та завантажувати їх у нашу базу даних, що є одним із ключових факторів.

Для зберігання даних ми обрали базу даних MySQL.

MySQL — це популярна система керування реляційними базами даних (RDBMS) з відкритим кодом, яка використовується для зберігання та керування структурованими даними. MySQL є однією з найпоширеніших баз даних у світі та використовується багатьма популярними веб-сайтами та програмами. MySQL базується на архітектурі клієнт-сервер, де клієнтська програма взаємодіє з сервером, який керує базою даних. Сервер може працювати на різних операційних системах, включаючи Linux, Windows і macOS.

Деякі з ключових функцій MySQL включають:

1. Масштабованість: MySQL може обробляти великі бази даних із мільйонами рядків і тисячами одночасних користувачів.

2. Безпека: MySQL підтримує функції шифрування, автентифікації та контролю доступу для забезпечення безпеки даних, що зберігаються в базі даних.

3. Продуктивність: MySQL оптимізовано для швидких операцій читання та запису та може обробляти високі навантаження трафіку.

4. Гнучкість: MySQL підтримує широкий діапазон типів даних, включаючи числові, рядкові формати та формати дати/часу, а також забезпечує підтримку різних мов програмування та API.

5. Продуктивність: MySQL має простішу архітектуру, що робить її швидшою та ефективнішою для простих програм, які інтенсивно читають. Він відомий своєю високою швидкістю та оптимізований для обробки великих наборів даних.

6. Простота використання: MySQL відносно легко налаштувати та використовувати, особливо для розробників, які знайомі з SQL. Його зручний інтерфейс і простий синтаксис роблять його ідеальним вибором для розробників, які тільки починають працювати з базами даних.

7. Популярність: MySQL є однією з найпопулярніших баз даних у світі з великою та активною спільнотою розробників. Така популярність означає, що існує велика кількість документації, навчальних посібників і ресурсів, які допоможуть розробникам усунути проблеми та оптимізувати свої бази даних MySQL.

8. Підтримка реплікації: MySQL чудово підтримує реплікацію бази даних, що дозволяє розробникам створювати кілька копій бази даних для резервного копіювання, балансування навантаження або відновлення після збоїв. Ця функція особливо корисна для програм високої доступності, яким потрібен постійний доступ до бази даних.

MySQL зазвичай використовується для веб-додатків, таких як системи керування вмістом, веб-сайти електронної комерції та онлайн-форуми. Він також використовується для сховищ даних, аналітики даних і програм бізнес-аналітики.

Для виконання розрахунків, пов'язаних з векторами та матрицями, ми використовували два пакети – NumPy та Numba. За документацією, NumPy – це один з основних пакетів, які вживаються для наукових обчислень в мові програмування Python. Цей пакет надає можливості для роботи з багатовимірними масивами, маскуваннями, матрицями та різними підпрограмами для швидких операцій над масивами, таких як математичні, логічні операції, сортування, вибір, введення/виведення, дискретні перетворення Фур'є. Крім того, цей пакет надає можливості для виконання операцій лінійної алгебри, базових статистичних операцій та моделювання випадкових величин.

У пакеті NumPy заснованому на об'єкті ndarray, є можливість працювати з масивами даних однакового типу в n-вимірному просторі. Багато операцій можуть бути виконані в скомпільованому коді на мові C++, що робить їх швидкими та ефективними. На відміну від стандартних послідовностей Python, масиви NumPy мають кілька важливих відмінностей [43]:

1. Масиви NumPy мають фіксований розмір при створенні на відміну від списків Python, які можуть динамічно зростати. Зміна розміру ndarray створить новий масив і видалить оригінал.

2. Всі елементи масиву NumPy повинні мати один і той же тип даних, а отже, однаковий розмір у пам'яті. Також елементами масиву можуть бути інші масиви, що дозволяє створювати матриці з різною кількістю стовпців або рядків.

3. Масиви NumPy створені за стандартами вдосконалених математичних та інших типів операцій з великою кількістю даних. Як

правило, такі операції виконуються ефективніше і з меншим кодом, ніж це можливо за допомогою вбудованих послідовностей Python.

4. Сьогодні масиви NumPy активно використовуються у науково-математичних пакетах Python, кількість яких інтенсивно зростає. Хоча масиви NumPy, як правило, підтримують введення послідовностей Python, пакети перетворюють такі послідовності в масиви NumPy перед обробкою. Іншими словами, для ефективного використання сучасного науково-математичного програмного забезпечення на базі Python потрібно також знати, як використовувати масиви NumPy.

Як вже згадувалося, існує ще один пакет для виконання матричних операцій – Numba [41]. Використання цього пакету дозволяє прискорити виконання операцій, що реалізовані в NumPy, тому що замість інтерпретації коду він використовує зібрану бінарну версію. Поєднання цих двох пакетів дозволяє отримати потужний інструмент для роботи з математичними об’єктами, який зберігає високорівневу абстракцію і має таку ж швидкодію, як алгоритми, написані на мовах програмування C++ або C.

Додатковою перевагою використання Numba є можливість паралельного обчислення, яке може бути виконане як на центральних процесорах, так і на графічних процесорах комп'ютерних систем. Використання обчислень на графічних процесорах (GPU) є важливим з кількох причин:

1. Графічні процесори мають багато ядер, які можуть виконувати багато операцій одночасно, що дозволяє прискорити обчислення.

2. Графічні процесори спеціалізовані на виконанні операцій зображення, таких як обробка зображень, графіка, відео та інші, тому вони мають високу продуктивність для таких операцій.

3. Використання графічних процесорів дозволяє відокремити обчислення від інших завдань на комп'ютері, що забезпечує більш ефективне використання ресурсів і покращує продуктивність.

Загалом використання обчислень на графічних процесорах є важливим для отримання швидких та ефективних результатів в галузі науки, техніки, медицини, фінансів та інших галузях, які потребують великі обчислювальні ресурси.

Для обробки даних та використання необхідних метрик ми обрали пакет sklearn [42].

Scikit-learn, або sklearn, є одним з найпопулярніших пакетів для машинного навчання (Machine Learning) в мові програмування Python. Він містить бібліотеку алгоритмів класифікації, регресії, кластеризації, пониження розмірності, відбору ознак та інших інструментів для розв’язання завдань машинного навчання.

Основні переваги sklearn полягають у наступному:

1. Простота використання та документація: Sklearn має дуже детальну документацію та набір прикладів, що дозволяє швидко і легко навчитися його використовувати. Інтерфейси бібліотеки складніші для розуміння у порівнянні з іншими популярними пакетами, такими як TensorFlow або PyTorch, проте він є дуже легким для початківців.

2. Широкий вибір алгоритмів: sklearn містить набір різних алгоритмів машинного навчання, таких як Random Forests, Gradient Boosting, Support Vector Machines, Neural Networks та інші. Кожен з цих алгоритмів має свої особливості та відповідає за розв’язання конкретної задачі.

3. Налагодження та оцінка моделей: sklearn надає інструменти для оцінки точності та налагодження параметрів моделей. Він містить методи оцінки, такі як метрики класифікації (accuracy, precision, recall, F1-score тощо), крос-валідація та валідація на відкладеному наборі даних.

4. Широкі можливості підготовки даних: sklearn містить інструменти для попередньої обробки даних, такі як масштабування, кодування категоріальних змінних, заповнення пропущених значень тощо.

Scikit-learn також має документацію високої якості та широку спільноту користувачів, що надає безкоштовну та відкриту підтримку. Додатково, пакет Scikit-learn підтримує інтеграцію з більшістю інших бібліотек Python, що забезпечує безперервну інтеграцію зі стеком технологій машинного навчання Python.

Ще однією перевагою Scikit-learn є те, що він здатний працювати з великою кількістю даних, має підтримку паралельної обробки та можливості розподілу обчислень, що дозволяє знизити час навчання моделей. Scikit-learn також має можливість оптимізації параметрів моделі, яка автоматично налаштовує їх на основі вхідних даних.

Для обробки природних мов (NLP) на Python використовують пакет spaCy, який є безкоштовною бібліотекою з відкритим кодом. Цей пакет розроблений спеціально для виробничого використання та допомагає створювати додатки, які обробляють та “розуміють” великі обсяги тексту. Він може бути використаний для побудови систем вилучення інформації або систем розуміння природної мови. Згідно з документацією програмного забезпечення [44], крім підтримки більше, ніж 64 мов та наявності 55 навчених конвеєрів для 17 мов, можливостями пакету також є:

1. Багатозадачне навчання з попередньо навченими трансформаторами, такими як BERT (bidirectional encoder representations from transformers);

2. Оброблені вектори слів;

3. Сучасна швидкість;

4. Токенізація з мовною мотивацією;

5. Компоненти для розпізнавання іменованих сутностей, позначення частини мови, розбору залежностей, сегментації речень, класифікації тексту, морфологічного аналізу, зв’язування сутності тощо;

6. Здатність легко розширюватися за допомогою нестандартних компонентів та атрибутів;

7. Підтримка спеціальних моделей у PyTorch, TensorFlow та інших фреймворках;

8. Вбудовані візуалізатори для синтаксису та NER.

У рамках кваліфікаційної роботи магістра ми будемо використовувати цей пакет для сіамських нейронних мереж, які будуть розраховувати схожість текстової інформації. Цей пакет також підтримує паралельні розрахунки, як і вищезгаданий комплекс пакетів.

Наступним пакетом, що використовується для збереження та обробки великого кластеру даних, є Pandas. Згідно з документацією програмного забезпечення [45] цей пакет дає велику кількість можливостей взаємодії з даними серед яких є можливості:

1. Швидко та ефективно звертатися до даних для подальшої маніпуляції;

2. Застосування інструментів для запису та зчитування даних з оперативної пам’яті у стандартизовані в усьому світі формати, як от CSV(comma separated values), Microsoft Excel, SQL-like databases, а також HDF5 формат;

3. Інтелектуального вирівнювання даних та інтегровану обробку відсутніх даних, що дає змогу отримати автоматичне вирівнювання на основі міток у обчисленнях та легко маніпулювати безладними даними в упорядкованій формі;

4. Гнучкішої зміни форми та обертання наборів даних;

5. Інтелектуальної «нарізки» на основі етикеток, химерне індексування та підмножина великих наборів даних;

6. Можливість вставляти стовпці та видаляти зі структур даних для змінності розміру;

7. Об'єднання або перетворення даних з потужною групою за допомогою механізму, що дозволяє розділити-застосувати-об'єднати операції над наборами даних;

8. Високопродуктивного злиття та об’єднання наборів даних у одну повноцінну структуру;

9. Ієрархічного індексування осей, що забезпечує інтуїтивний спосіб роботи з великими розмірами даних у нижчій розмірній структурі даних;

10. Застосування функціоналу часових рядів: генерація діапазону дат та перетворення частоти, статистика переміщення вікон, зміщення та відставання дат, створення часових зсувів для конкретного домену та приєднання до часових рядів, не втрачаючи даних тощо.

Такий набір характеристик робить цей пакет оптимальним для використання у розв’язанні поставленої задачі.

Для відображення результатів роботи системи при різних конфігураціях та на різних навчальних вибірках часто використовують певну графічну візуалізацію. У цій сфері широко використовується пакет Matplotlib, що є бібліотекою для візуалізації даних та побудови графіків для Python та розширення NumPy. Цей пакет є життєздатною альтернативою з відкритим кодом до MATLAB.

З документації програмного забезпечення [46] дізнаємось, що пакет matplotlib побудований таким чином, що кілька рядків коду – це все, що потрібно в більшості випадків для створення візуального сюжету даних. Шар сценарію matplotlib складається з двох API:

1. API pyplot – це ієрархія об’єктів коду Python, огорнута інтерфейсом matplotlib.pyplot;

2. Колекція об’єктів API (об’єктно-орієнтованого), які можна зібрати з більшою гнучкістю, ніж pyplot. Цей API забезпечує прямий доступ до серверних шарів Matplotlib.

Інтерфейс Pyplot API має зручний стиль, схожий на MATLAB. Оригінальна мета розробки matplotlib полягала в створенні відкритої альтернативи для MATLAB. API MATLAB та його інтерфейс є більш потужним та гнучким, ніж pyplot, але натомість вважається складнішим для використання. Для роботи з графіками розуміння API matplotlib pyplot є ключовим. Розглянемо кілька інтерфейсів взаємодії:

1. matplotlib.pyplot.figure – контейнер верхнього рівня, який містить все, що візуалізується в сюжеті, включаючи одну або кілька вісей координат;

2. matplotlib.pyplot.axes – це вісі координат, які містять більшість елементів у графіку: Axis, Tick, Line2D, Text тощо.

Також цей пакет дозволяє будувати будь-які види аналітичних графіків, як от гістограми, коробка з вусами, матриці збурень, тощо.

3.2 Вибір методу оцінювання якості

Наразі існує багато метрик, які оцінюють якість роботи рекомендаційних систем, як от:

1. Precision – це відношення кількості релевантних рекомендацій до загальної кількості рекомендацій, що були надіслані користувачу.

2. Recall – це відношення кількості релевантних рекомендацій до загальної кількості релевантних елементів у системі.

3. F1-score – це середнє гармонійне значення точності та повноти.

4. MAP (Mean Average Precision) – це середнє значення точності відповідних документів, що повинні бути вище ранжировані в результаті пошуку.

5. NDCG (Normalized Discounted Cumulative Gain) – це метрика, що оцінює ранжування, звертаючи увагу на те, наскільки релевантні елементи знаходяться вгорі списку рекомендацій.

6. ROC-AUC (Receiver Operating Characteristic - Area Under Curve) – це метрика, що використовується для оцінки ранжування, коли потрібно вибрати поріг на підставі ймовірності.

7. RMSE (Root Mean Square Error) – це метрика, яка використовується для оцінки точності рейтингів елементів в колаборативних системах.

8. MAE (Mean Absolute Error) – це метрика, яка використовується для оцінки середньої абсолютної помилки у рейтингах елементів.

Ці метрики можуть бути використані як для оцінки ефективності рекомендаційних систем в цілому, так і для оцінки їх окремих компонентів, таких як алгоритми ранжування, стратегії зважування, моделі рейтингування тощо.

Найбільш підходящі метрики залежать від конкретного типу рекомендаційної системи, її цілей та метрик користувачів, які є ключовими для певного додатку. Наприклад, у випадку рекомендаційних систем музичних композицій, метрики, які оцінюють точність рейтингів, можуть бути більш підходящими, ніж метрики, які оцінюють ранжування. На відміну від цього, для електронної комерції можуть бути важливими метрики, що враховують ранжування та точність.

Крім того, важливо брати до уваги не тільки значення метрик, але й контекст, в якому вони були отримані. Наприклад, низькі значення метрик можуть свідчити про неефективність рекомендаційної системи, але також можуть свідчити про те, що потрібно врахувати додаткові чинники, які можуть впливати на рекомендації.

У будь-якому випадку, при використанні метрик важливо збирати дані та аналізувати результати, щоб зрозуміти, які зміни можуть бути введені для поліпшення ефективності рекомендаційної системи.

Розглянемо більш детально кожний із зазначених методів оцінювання.

Метрика оцінювання Precision зазвичай використовується у рекомендаційних системах та машинному навчанні. Вона визначається як відношення кількості правильно відібраних позитивних відгуків до загальної кількості відібраних позитивних відгуків та негативних відгуків та має формулу такого вигляду:

$$precision = \frac{TP}{TP+FP}$$

де TP (True Positive) – кількість правильно відібраних позитивних відгуків, FP (False Positive) – кількість помилково відібраних позитивних відгуків.

Значення Precision може бути в діапазоні від 0 до 1, де 1 вказує на ідеальну точність. Вищі значення Precision свідчать про те, що система рекомендацій правильно відбирає релевантні результати для користувачів.

Precision є особливо важливою метрикою в ситуаціях, коли помилкові відгуки мають великий вплив на користувача. Наприклад, у випадку рекомендаційних систем для медичних діагнозів, помилкові відгуки можуть мати серйозні наслідки, тому важливо забезпечити високу точність (Precision) системи рекомендацій.

Проте, слід пам'ятати, що Precision має деякі обмеження. Наприклад, він не враховує кількість помилково не відібраних позитивних відгуків, що може призвести до переоцінки реальної ефективності системи рекомендацій. Також, Precision не враховує ранжування результатів, тобто він не вказує на те, наскільки релевантні результати відсортовані відповідно до їхньої важливості для користувача.

Метрика Recall є ще однією важливою метрикою якості в рекомендаційних системах та машинному навчанні. Вона визначається як відношення кількості правильно відібраних позитивних відгуків до загальної кількості позитивних відгуків та має формулу такого вигляду:

$$recall = \frac{TP}{TP+FN}$$

де TP (True Positive) – кількість правильно відібраних позитивних відгуків, FN (False Negative) – кількість помилково не відібраних позитивних відгуків.

Значення Recall також може бути в діапазоні від 0 до 1, де 1 вказує на ідеальну повноту. Вищі значення Recall свідчать про те, що система рекомендацій правильно відбирає усі релевантні результати для користувачів.

На відміну від Precision, Recall зосереджується на віднесенні всіх релевантних результатів до загальної кількості позитивних відгуків. Це особливо важливо в ситуаціях, коли пропуск релевантних результатів є більш серйозним, ніж вказування неправильних результатів.

Наприклад, у випадку системи рекомендацій для виявлення підозрілих транзакцій на банківському рахунку, Recall є більш важливою метрикою, ніж Precision. У такому випадку, помилкова ідентифікація транзакції як підозрілої може призвести до додаткових перевірок, але пропуск підозрілої транзакції може призвести до фінансових втрат.

Метрика F1-score (також відома як F1-measure) є комбінованою метрикою, що враховує як метрику Precision, так і метрику Recall. Вона використовується для оцінки якості системи рекомендацій та вважається більш інформативною, ніж окремі метрики Precision або Recall.

F1-score обчислюється як гармонічне середнє Precision і Recall. Гармонічне середнє дозволяє більш точно врахувати значення, що є близькими до нуля:

$$F1 = \frac{2 * (preceision * recall)}{precision + recall}$$

Значення F1-score може бути від 0 до 1, де 1 вказує на ідеальну систему рекомендацій. Значення менше 1 вказує на те, що є проблеми з якістю рекомендаційної системи.

F1-score дозволяє врахувати як точність (Precision), так і повноту (Recall) системи рекомендацій одночасно. Значення F1-score покращується, якщо система рекомендацій може правильно відібрати більше позитивних результатів і зменшити кількість помилково відібраних результатів.

Метрика MAP (Mean Average Precision) є метрикою, яка використовується для оцінки якості систем рекомендацій в задачах ранжування результатів. Вона враховує не тільки правильність рекомендації, але і їх порядок.

MAP обчислюється шляхом взяття середнього значення Precision для кожної позиції у списку рекомендацій, до якої було досягнуто певного рівня Recall. Інакше кажучи, MAP вимірює, наскільки часто коректна рекомендація знаходиться на вищій позиції у списку порівняно з некоректною.

Оцінка MAP зазвичай розраховується за допомогою такої формули:

$$MAP = \frac{1}{|U|}\sum_{i=1}^{|U|}\left(\frac{1}{n_i}\right)\sum_{j=1}^{n_i} P_j$$

де |U| – множина кристувачів, n_i – кількість релевантних об'єктів для $i-$го користувача, P_j – precision для $j-$ї позиції у списку рекомендацій $i-$го користувача.

Значення MAP може бути від 0 до 1, де 1 означає ідеальну систему рекомендацій, а 0 – систему, яка не дає жодних коректних рекомендацій. MAP краще використовувати для порівняння різних систем рекомендацій з однаковою кількістю рекомендацій, оскільки він залежить від кількості рекомендацій для кожного користувача. Також слід враховувати, що MAP дуже чутливий до невеликих змін у ранжуванні, тому його важливо розглядати разом з іншими метриками, такими як Precision та Recall.

Ще один метод оцінювання nDCG використовується з урахуванням таких припущень:

1. Документи з високою релевантністю повинні з'являтися на початку списку.

2. Функція релевантності змінює своє значення від 0 до $n > 1$.

Логіка методу полягає у тому, що документи з високою релевантністю, які потрапили на низькі позиції повинні, "штрафуватися", тому відповідні ваги зменшуються логарифмічно пропорційно позиції у списку. Оскільки значення методу не є обмеженим згори, то зазвичай використовують таку міру:

$$nDCG = \frac{DCG_P}{IDCG_P}$$

де величина $DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)}$, а величина $IDCG_p$ обраховується за тим самим правилом, що і DCG_p, але список є відсортованим за релевантністю, що дає можливість обмежити значення отриманої величини. Сфера застосування цього методу аналогічна до попередніх.

Метрика RMSE (Root Mean Square Error) – це метрика, яка використовується для вимірювання точності регресійних моделей. Вона вимірює середнє квадратичне відхилення прогнозів моделі від фактичних значень.

RMSE визначається наступним чином: спочатку обчислюється різниця між прогнозованими та фактичними значеннями, після чого отримані значення підносяться до квадрату, далі обчислюється середнє значення цих квадратів, а наостанок виконується взяття квадратного кореня.

Формально RMSE визначається наступним чином:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

де Y – фактичне значення, $\widehat{Y}$ – прогнозоване значення, n - кількість спостережень. Чим менше значення RMSE, тим більш точна модель. RMSE добре працює, коли помилки моделі рівномірно розподілені та нормально розподілені, але він не рекомендується використовувати, коли помилки моделі мають складний вигляд або виявляють систематичну залежність від змінних.

MAE (Mean Absolute Error) – це метрика, яка використовується для вимірювання точності регресійних моделей. Вона вимірює середню абсолютну різницю між прогнозованими і фактичними значеннями.

MAE визначається наступним чином: спочатку обчислюється різниця між прогнозованими та фактичними значеннями, далі отримані значення беруться за модулем, після чого обчислюється середнє значення отриманих модулів.

Формально MAE визначається наступним чином:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

де Y – фактичне значення, $\widehat{Y}$ – прогнозоване значення, n - кількість спостережень.

MAE є більш робастною метрикою, ніж RMSE, оскільки не залежить від значень відхилень із сильним впливом на результат, тому вона часто використовується, коли потрібно оцінювати точність моделей з великими значеннями відхилень. Однак MAE менш чутлива до великих відхилень, тому може не давати точного результату при оцінці моделей з розподілом помилок, що мають важку хвостову частину.

Наша робота полягає в оцінці ефективності алгоритму узагальнення, тому ми вважаємо, що метод крос-валідації є найбільш відповідним. Цей метод полягає у тому, щоб розділити вибірку даних на дві групи – тренувальну та валідаційну, і розглядати кожну групу окремо.

Тренувальна група дозволяє правильно налаштувати параметри системи для максимальної ефективності та перевірити її функціональність. Зазвичай, для автоматичного налаштування параметрів системи використовують алгоритми оптимізації гіперпараметрів, які дозволяють значно спростити та прискорити процес. Ці алгоритми дозволяють одночасно перевіряти кілька конфігурацій системи за допомогою паралельних обчислень.підбору

Існує безліч алгоритмів для добору параметрів алгоритмів. Розглянемо деякі з найбільш популярних і широко використовуваних алгоритмів для підбору параметрів алгоритму.

1. Grid Search: Grid Search – це простий алгоритм підбору параметрів, де ми вибираємо набір значень параметрів та перебираємо їх усі можливі комбінації. Це робиться для знаходження найкращої комбінації параметрів, що дає найкращий результат.

2. Random Search: Random Search – це алгоритм, який генерує випадкові комбінації параметрів і оцінює їх ефективність, щоб знайти найкращу комбінацію параметрів.

3. Bayesian Optimization: Bayesian Optimization – це метод, який використовує статистичні методи для моделювання функції витрат та підбору параметрів, щоб знайти найкращі значення параметрів [19].

4. Genetic Algorithm: Генетичний алгоритм – це алгоритм, який використовує механізм еволюції для добору параметрів. Він працює шляхом створення випадкових комбінацій параметрів та оцінює їх ефективність. Потім він застосовує генетичні операції, такі як схрещування та мутація, щоб створити нащадки з найкращими комбінаціями параметрів.

Ці алгоритми можуть бути розділені на наступні категорії:

1. Детерміновані – це алгоритми, які забезпечують однаковий результат при кожному запуску на однаковому наборі даних. Наприклад, Grid Search - детермінований алгоритм.

2. Стохастичні (випадкові) – це алгоритми, які використовують випадковість для генерації нових комбінацій параметрів.

3. Засновані на моделюванні – це алгоритми, які використовують математичну модель для підбору параметрів. Наприклад, Bayesian Optimization використовує модель для навчання та передбачення результатів на основі раніше оцінених комбінацій параметрів.

4. Популяційні (генеративні) – це алгоритми, які використовують популяцію комбінацій параметрів для підбору оптимальних параметрів. Наприклад, Genetic Algorithm використовує популяцію комбінацій параметрів для знаходження найкращої комбінації.

Кожен з алгоритмів має свої переваги та недоліки, які можна порівняти. Ось деякі загальні висновки щодо кожного алгоритму:

1. Grid Search:

а) переваги: простий у використанні, забезпечує гарантоване знайдення оптимальних параметрів, особливо якщо рівень гіперпараметрів відомий.

б) недоліки: може бути дуже часово витратним, особливо якщо гіперпараметрів дуже багато. Не рекомендується для задач з великою кількістю гіперпараметрів.

2. Random Search:

а) переваги: простий у використанні, може знайти оптимальні параметри дуже швидко, особливо якщо знання про простір гіперпараметрів невелике.

б) недоліки: може знайти менш оптимальні параметри, особливо якщо простір гіперпараметрів є вузьким або невідомим.

3. Model-based algorithms:

а) переваги: ефективність залежить від якості моделі, здатність пристосовуватися до складних функцій, можливість знаходити оптимальні параметри швидко.

б) недоліки: потребують великої кількості обчислень, особливо якщо модель складна, можуть бути дуже чутливі до випадкових помилок, відсутність можливості знаходити глобальний оптимум.

4. Population-based algorithms:

а) переваги: здатність знаходити глобальний оптимум, можливість пристосовуватися до складних функцій, здатність знаходити оптимальні параметри в умовах обмежень на ресурси.

б) недоліки: потребують значної кількості обчислень, можуть бути дуже чутливі до початкових параметрів, можуть страждати від проблеми попадання в локальні мінімуми.

Загалом можна сказати, що алгоритми, які здатні адаптуватися до складних функцій та знаходити глобальний оптимум, мають більше переваг ніж недоліків. Тому, якщо задача вимагає знаходження оптимальних параметрів у складній функції, бажано використовувати Model-based та Population-based алгоритми.

Якщо ж задача є менш складною та кількість гіперпараметрів не є дуже великою, то Grid Search та Random Search можуть бути більш підходящими варіантами.

На нашу думку, найкращим алгоритмом для пошуку оптимальної конфігурації у контексті цієї роботи, є алгоритм Байєсівської оптимізації, оскільки кількість параметрів, що можуть бути змінені, достатньо невелика.

Суть *валідуючої групи* полягає у тому, щоб перевірити наскільки система була якісно налаштована та відповісти на питання "наскільки отриманий результат відхиляється від реального" для подальшого аналізу, який може покращити якість роботи системи. Для цього найчастіше використовують такі метрики:

- середньоквадратичне відхилення:

$$p(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} = RMSE$$

- середнє абсолютне відхилення:

$$p(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| = MAE$$

Розберемо умовні позначення у вищезазначених формулах: n – розмірність виходу системи; Y_i – і-й елемент вектору розмірності n, який складається з реальних спостережень, у випадку тематики, що розглядається, – це оцінка фільму; $\hat{Y}_i$ – і-й елемент вектору аналогічної розмірності, який складається з виходів системи (у випадку колаборативної фільтрації – це оцінка, яку б користувач міг би теоретично поставити об'єкту).

Легко бачити, що чим менша величина $p\ (y,\ \hat{y}\ )$, тим точнішою є апроксимована системою оцінка .

Розглянемо процедуру валідації роботи побудованої системи. Оскільки початковий набір даних є лише витягом “сухих” (таких, що не розділені на валідовані та тестові групи) даних, то необхідно буде зробити деякі модифікації з ними, щоб в подальшому застосувати для методу cross-validation.

Оберемо, наприклад, $TU\ (test\ users)$ користувачів з загальної вибірки. Далі для кожного об'єкта з історії переглядів користувача розрахуємо його оцінку на основі $LV\ (last\ viewed)$ попередньо переглянутих об'єктів. Далі для кожного користувача з вибірки розрахуємо відхилення апроксимованої оцінки від реальної. Як результат отримаємо набір відхилень спираючись на кількість використовуємих об'єктів для оцінювання.

3.3 Аналіз результатів

Розглянемо якість роботи наївного методу колаборативної фільтрації та узагальненного методу колаборативної фільтрації з використанням часового фактору та семантичної подібності. Результати були отримані на двох вибірках даних, а саме музиці та фільмах, як приклади того, що алгоритм можна застосовувати у різних галузях.

Обидва набори даних мали зазначену у розділі 2.1 обмеження, а саме:

1) Часова інформація, в нашому випадку дата виставлення оцінки користувачем

2) Оцінки, які користувачі виставили відповідним об'єктам

3) Текстова інформація, у випадку фільмів це біли назви стрічок та їх опис, а у випадку пісень, це були назви на тексти пісень.

Нижче нами будуть наведені результати отриманих похибок при апроксимації оцінок користувачів. Пояснимо сенс кожного з наведених графіків. По вертикалі в нас відображені помилки обчислень, а по горизонталі – унікальні ідентифікатори об'єктів, для яких ми апроксимували оцінки. Чим меншим стовпчик є за розміром, тим меншою є похибка обчислень. Також можна побачити стовпчики різних кольорів: синій та помаранчевий. Синім кольором відображаються похибки запропонованого нами методу, а помаранчевим – похибки наївного методу колаборативної фільтрації.

Наведемо приклади похибок при апроксимації оцінок для фільмів:

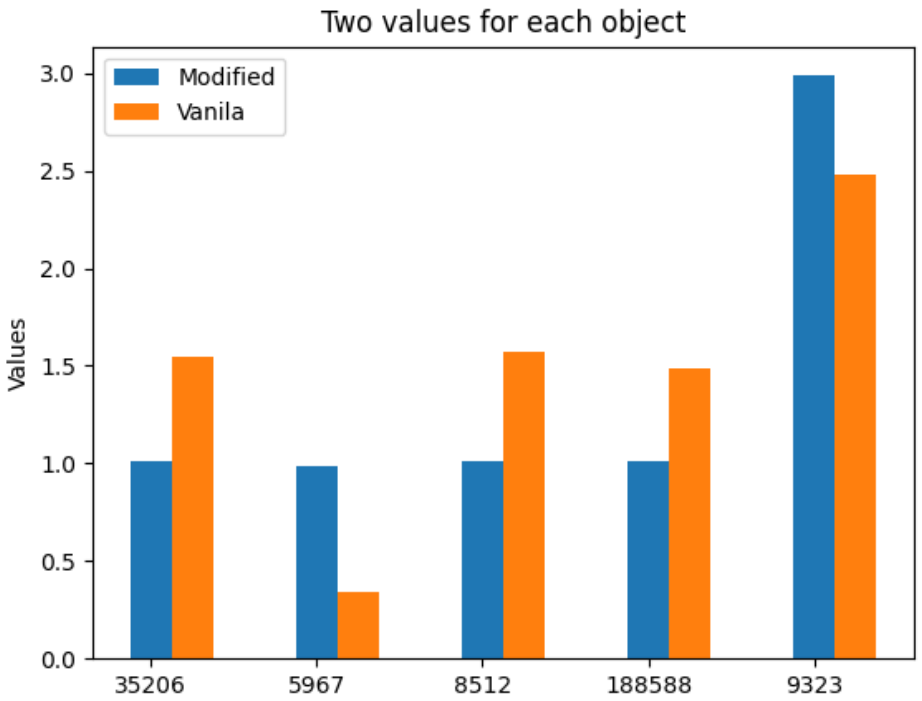

Рисунок 3.1

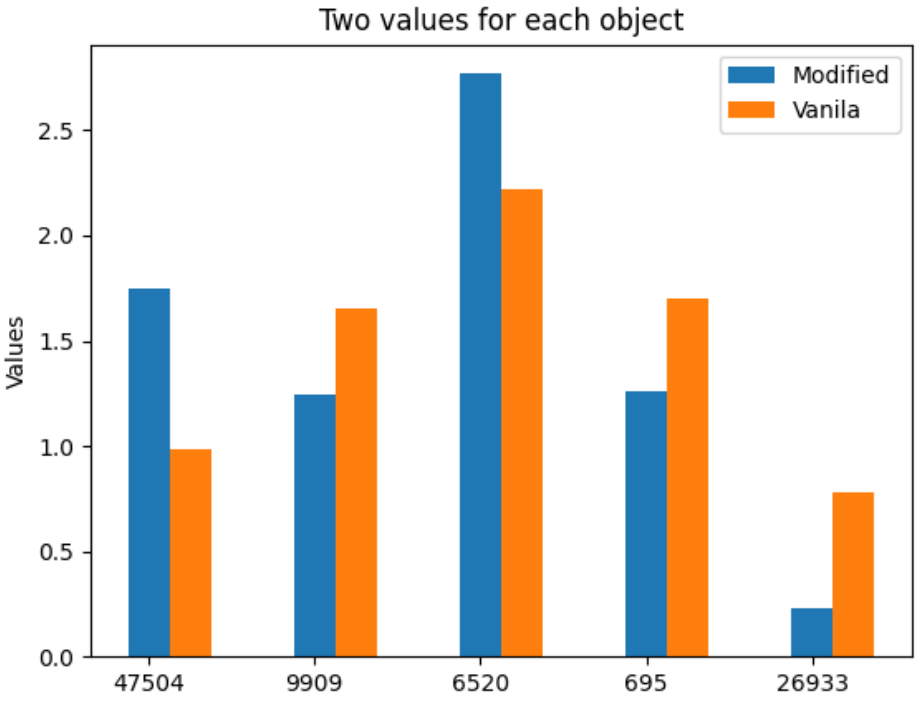

Рисунок 3.2

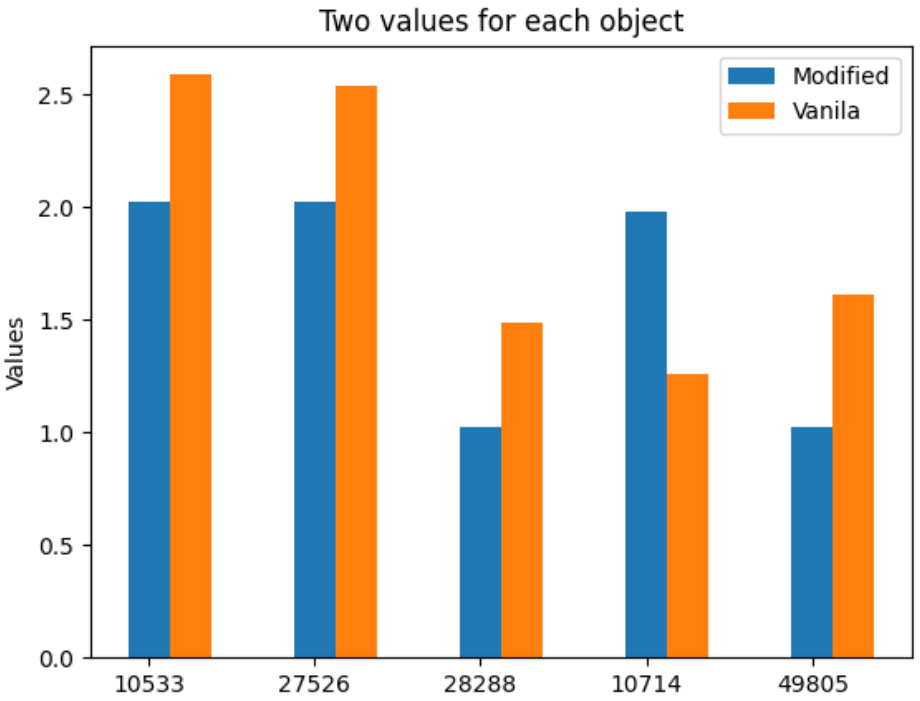

Рисунок 3.3

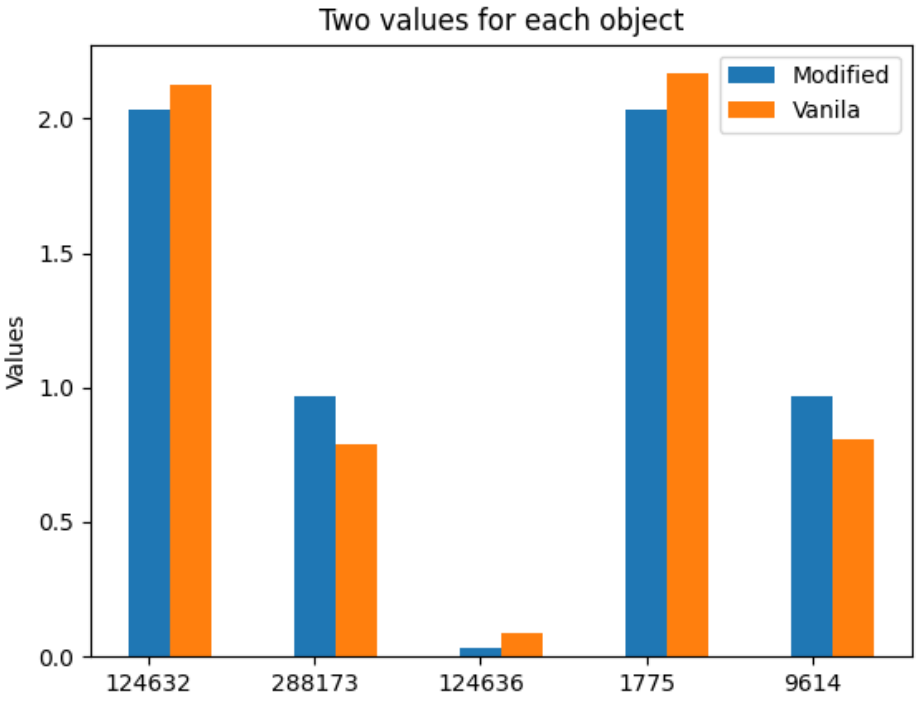

Рисунок 3.4

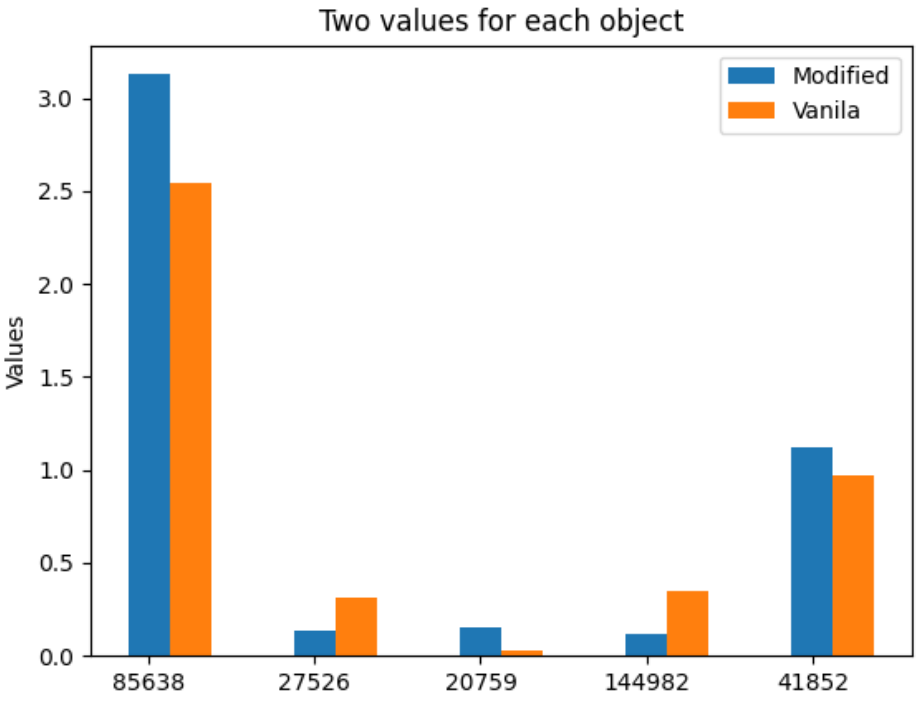

Рисунок 3.5

Також наведемо приклади похибок при апроксимації оцінок для музикальних стрічок:

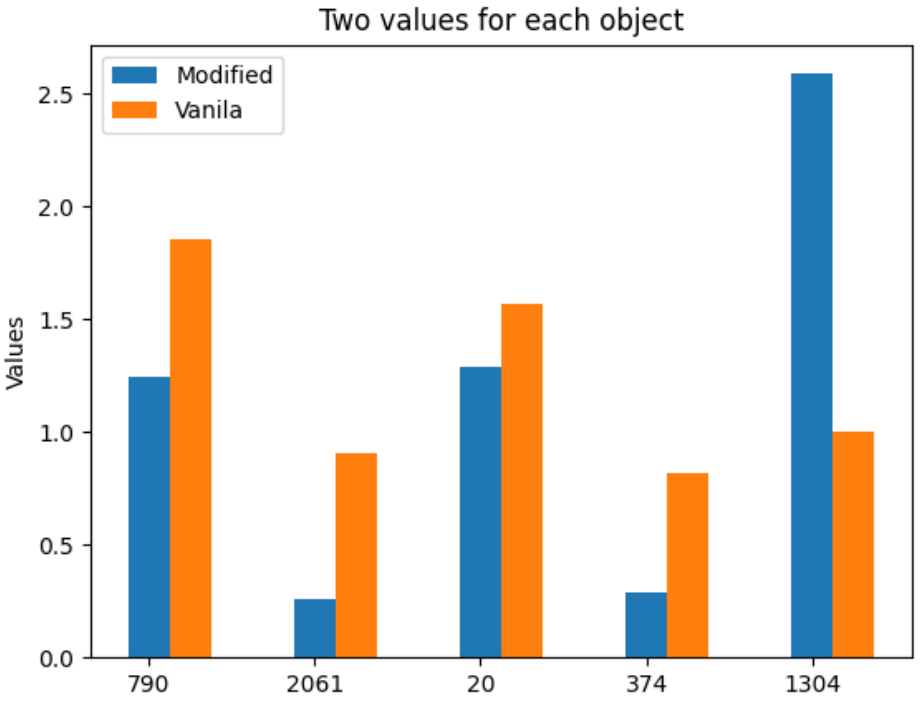

Рисунок 3.6

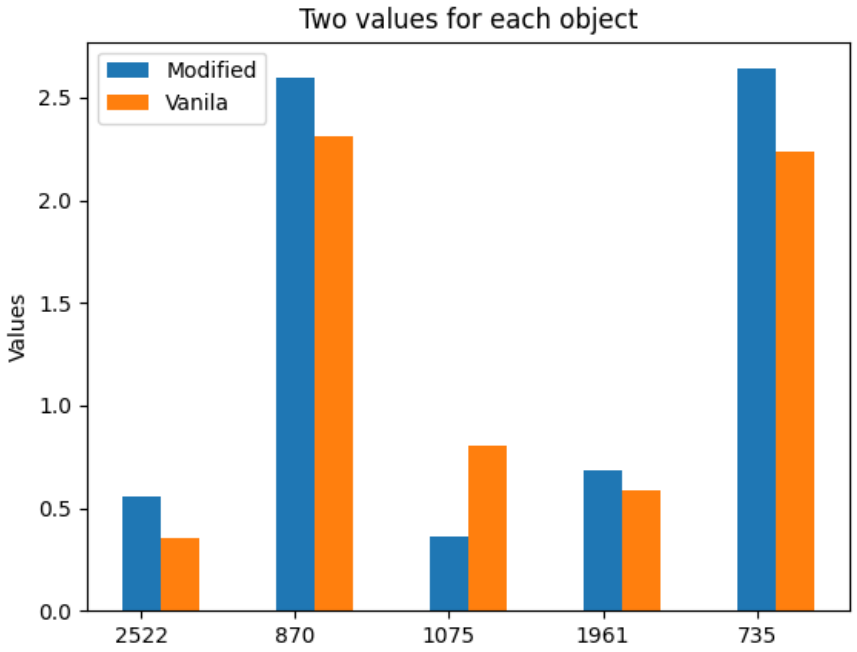

Рисунок 3.7

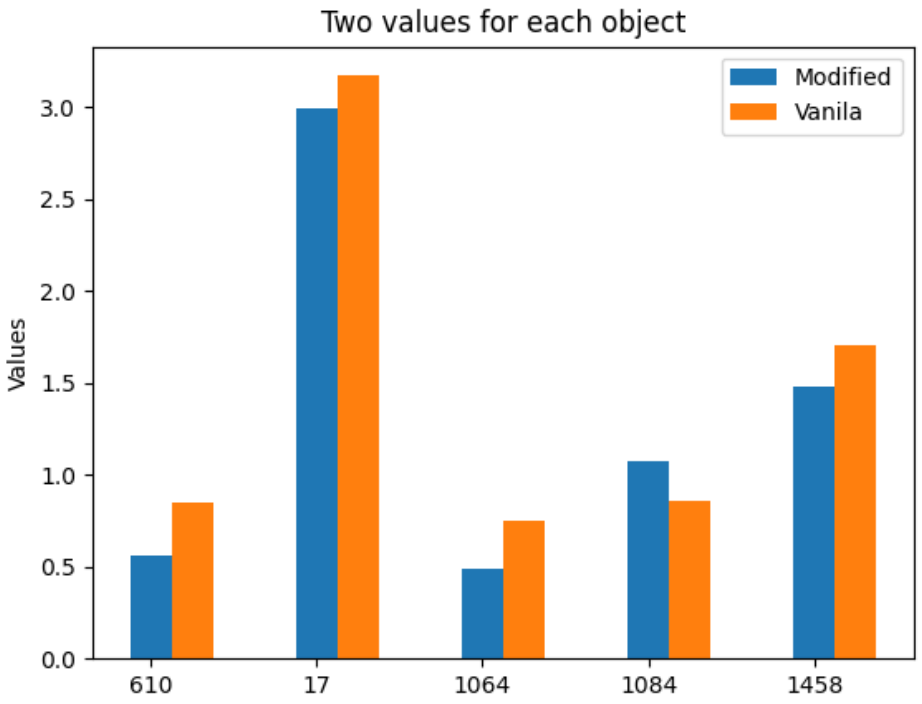

Рисунок 3.8

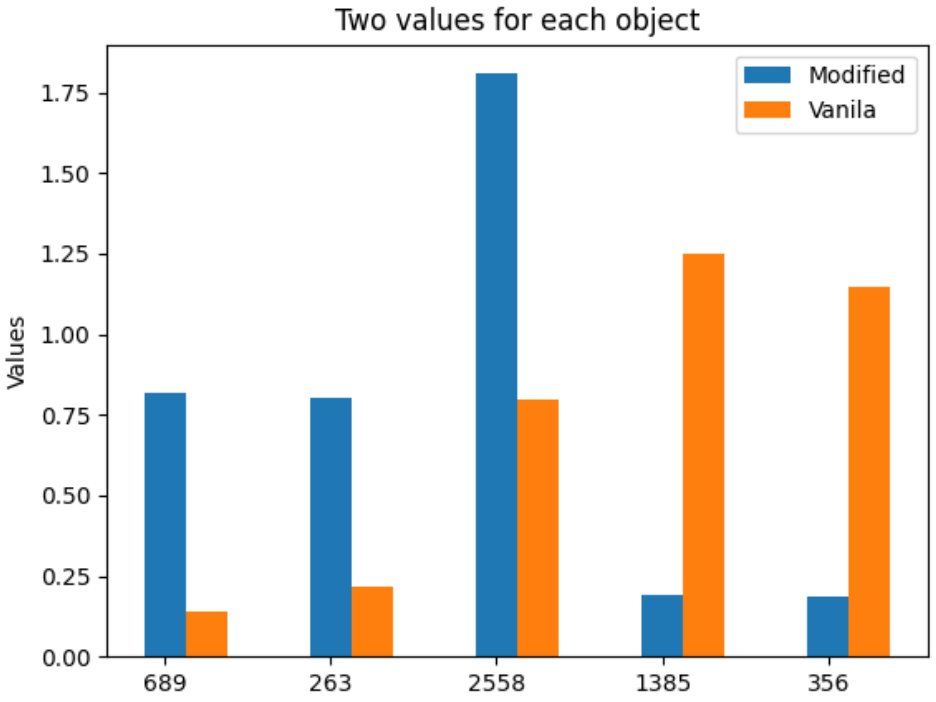

Рисунок 3.9

Також наведемо графік середньої швидкості обчислень відповідних апроксимацій:

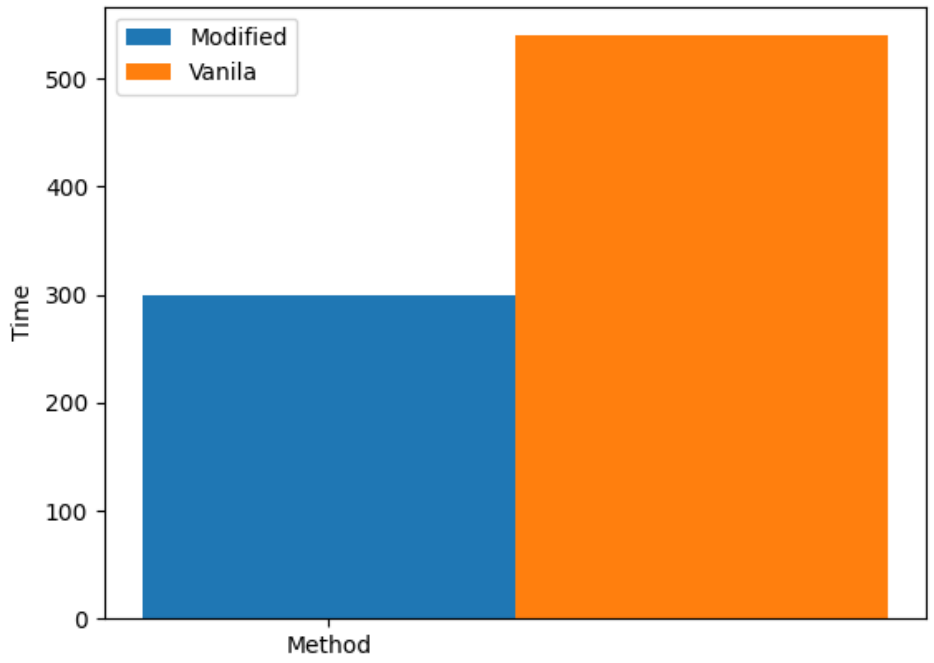

Рисунок 3.10

Отже як можна бачити, здебільшого узагальнений алгоритм працює краще, ніж його немодифікована версія. Також треба зазначити, що і час, який витрачається на обрахунки, також зменшився.

ВИСНОВКИ

В нашій роботі був розглянутий та проаналізований наївний метод колаборативної фільтрації типу item-based. За визначенням він дозволяє не лише зробити оцінку більш “стійкою” до збурень, оскільки на відміну від інтересів користувачів, які можуть постійно змінюватись, середня оцінка об'єкта залишається постійною, а також пришвидшити обробку даних.

Ми узагальнили метод колаборативної фільтрації з використанням часового та семантичних факторів шляхом введення відповідних обмежень на вхідні дані та їх попередньої обробки, що дало змогу використовувати цю модифікацію у різних доменних областях.

Нами було розроблено відповідне програмне забезпечення, під час роботи над яким я вивчив відповідні пакети прикладного програмного забезпечення – HDBSCAN, Matplotlib, Numpy, Numba, Pandas, SpaCy, SQL – і навчився ними користуватися.

Ми також перевірили адекватність роботи запропонованого нами методу використовуючи набори даних з різних доменних областей.

У результаті перевірки нами було виявлено той факт, що наша модифікація працює краще, ніж наївний метод, при чому у будь-якій з доменних областей до 20%.

Отже, можна стверджувати, що в результаті проведеної роботи мету було досягнуто, поставлені завдання були виконані.

Запропоноване узагальнення методу може знайти своє застосування у сервісах інтернет-речей, будь то стрімінговий сервіс, інтернет магазин тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Альошин І., Бородіна А., Кікоть І. Особливості програмування та нові можливості програмування PYTHON. – [Електронний ресурс] / І. Альошин, А. Бородіна, І. Кікоть // Системи управління, навігації та зв'язку, 2018. – 4 № 50. – Режим доступу: http://journals.nupp.edu.ua

2. Альперт С.І. Основні міри подібності та нові підходи до їх застосування при класифікуванні гіперспектральних космічних зображень / С.І. Альперт. – Access Path:http://www.immsp.kiev.ua/publications/articles/2019/2019_1

3. Базурін В., Омелечко Є., Ковтун А. Порівняльний аналіз середовищ програмування мовою Python. – [Електронний ресурс] / В. Базурін, Є Омелечко, А. Ковтун // Інформаційно-комунікаційні технології в освіті. – С. 281-291. – Режим доступу: https://lib.iitta.gov.ua

4. Копей В. Б. Мова програмування Python для інженерів і науковців: Навчальний посібник. – [Електронний ресурс] / В. Б. Копей. – Івано-Франківськ : ІФНТУНГ, 2019.. – 274с. – Режим доступу: https://chtyvo.org.ua

5. Мелешко Є. В. Проблеми сучасних рекомендаційних систем та методи їх рішень. – [Electronic Resource] / Мелешко Є.В. // Системи управління, навігації та зв'язку, 2018. – 4 (50). – Access Path: http://www.irbis-nbuv.gov.ua

6. Меньшикова Н. В. Портнов И.В. , Николаев И.Е. Обзор рекомендательных систем и возможностей учета контекста при формировании индивидуальных рекомендаций. – [Electronic Resource] / Н. В. Меньшикова, И.В. Портнов, И.Е. Николаев. // ACADEMY, 2016. – No 6. – С. 20–22. – Access Path: https://cyberleninka.ru/article

7. Чередніченко О., Іващенко О., Гонтар Ю, Ворона Б. Інтелектуальний аналіз пропозицій товарів на основі контекстних рекомендацій [Электронный ресурс] / О. Чередніченко, О. Іващенко, Ю. Гонтар, Б. Ворона // Вісник Національного технічного університету «ХПІ». Серія:

Системний аналіз, управління та інформаційні технології. – № 44 (1320). – 2018. – Режим доступу: http://library.kpi.kharkov.ua/files/Vestniki/2018_44

8. Adomavicius G., Tuzhilin A. Context-aware recommender systems. – [Electronic Resource] / G. Adomavicius, A. Tuzhilin / Recommender systems handbook. – Springer, 2011. – P. 217–253. – Access Path: https://www.researchgate.net

9. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. – [Electronic Resource] / G. Adomavicius, A. Tuzhilin // Knowledge and Data Engineering, IEEE Transactions. –2005. – V.17(6). – P. 734–749. – Access Path: https://www.researchgate.net

10. Aggarwal C. C. Recommender Systems. – [Электронный ресурс] / C. C. Aggarwal // Springer. — 2016. – 518 p. — Access Path: https://www.springer.com

11. Ankerst M., Breunig M.M., Kriegel H.P., Sander J. Optics: ordering points to identify the clustering structure. – [Електронний ресурс] / Ankerst M., Breunig M.M., Kriegel H.P., Sander J. // Proceedings of International Conference on Management of Data, 1999. – 28 . – P. 49–60. – Access Path: https://www.researchgate.net

12. Berkovsky S., Cantador I., Tikk D. Collaborative recommendations: algorithms, practical challenges and applications. – [Электронный ресурс] / S. Berkovsky, Cantador I., D. Tikk // World scientific publishing, 2019. – Access Path: www.amazon.com/Collaborative-Recommendations-Algorithms-Chall

13. Bi Y., Song L., Yao M., Wu Z., Wang J., Xiao J. DCDIR: a deep cross-domain recommendation system for cold start users in insurance domain. – [Electronic Resource] / Bi Y., Song L., Yao M., Wu Z., Wang J., Xiao J.// Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020. – Access Path: https://arxiv.org/abs/2007.13316

14. Brangbour E., Bruneau P., Tamisier T., Marchand-Maillet S. Active learning with crowdsourcing for the cold start of imbalanced classifiers . – [Electronic Resource] // Lecture notes in computer science. Springer International Publishing, 2020. – P. 192–201. – Access Path: https://www.researchgate

15. Bromley J. , Guyon I. , LeCun Y. , Sickinger E. Signature Verification using a "Siamese" Time Delay Neural Network. – [Електронний ресурс] – 1993 . – Access Path: https://proceedings.neurips.cc/paper

16. Burke R. Hybrid Web Recommender Systems . – [Electronic Resource] / Burke R.. // The Adaptive Web. Lecture Notes in Computer Science. – vol 4321. Springer, Berlin, Heidelberg, 2007. – C. 377–408. – Access Path: https://www.researchgate.net/publication

17. Desrosiers C., Karypis G. A comprehensive survey of neighborhood based recommendation methods. – [Electronic Resource] / [Desrosiers C., Karypis G.] / Recommender systems handbook. – Springer, 2011. – P.107–144. – Access Path: https://link.springer.com/chapter

18. Ester M., Kriegel H.P., Sander J., Xu. X. A density-based algorithm for discovering clusters in large spatial databases with noise. – [Електронний ресурс] / Ester M., Kriegel H.P., Sander J., Xu. X. // Proceedings of the Int. Conf. Knowl. Discovery and Data Mining, 1996. – Access Path:https://www.semanticscholar.org/paper

19. Frazier P. A tutorial on Bayesian optimization. – [Electronic Resource] / P. Frazier // Cornell University, 2018. – Access Path: https://arxiv.org/abs/1807.02811

20. Goldberg D, Nichols D, Oki BM, Terry D. Using collaborative filtering to weave an information tapestry. – [Electronic Resource] / Goldberg D, Nichols D, Oki BM, Terry D.// Commun ACM, 1992. – 35.– P. 61–70. – Access Path:https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference

21. Goldberg Y. A Primer on Neural Network Models for Natural Language Processing. – [Electronic Resource] / Goldberg Y. |// Journal of Artificial

Intelligence Research, 2016. – 57. – C. 345–420. – Access Path: https://www.researchgate.net/publication

22. Hinneburg A., Keim D.A. A general approach to clustering in large databases with noise. – [Електронний ресурс] / Hinneburg A., Keim D.A. // Knowledge and Information Systems, 2003. – 5 (4). – P. 384-415. – Access Path: https://www.researchgate.net/

23. Isaac A.G.Simulating Evolutionary Games: a Python-Based Introduction 2008. – [Електронний ресурс] / Isaac A.G. // Journal of Artificial Societies and Social Simulation, 2008. – 11(38). – Access Path: http://jasss.soc.surrey.ac.uk

24. Kusner, M. J., Sun, Y., Kolkin, N. I., Weinberger, K. Q. From word embeddings to document distances. – [Electronic Resource] / Kusner, M. J., Sun, Y., Kolkin, N. I., Weinberger, K. Q. // Proceedings of the 32nd International Conference on Machine Learning, 2015. – Vol. 37. – P. 957-966. – Access Path: https://proceedings.mlr.press/v37/kusnerb15.html

25. Kyung-Hyan Yoo, Gretzel U. Creating More Credible and Persuasive Recommender Systems: The Influence of Source Characteristics on Recommender System Evaluations. – [Electronic Resource] / [Yoo Kyung-Hyan, U. Gretzel] / Recommender systems handbook. – Springer, 2011. – P. 455–477. – Access Path: https://www.researchgate.net

26. Marappan R. Recommender System for Movielens Datasets using an Item-based Collaborative Filtering in Python. – [Електронний ресурс] / Marappan R. // International Journal of Mathematical, Engineering, Biological and Applied Computing, 2022. – 1(1). – P. 42–43. – Access Path: https://www.scipublications.com/journal

27. Natarajan S., Vairavasundaram S., Natarajan S., Gandomi A.H. Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data.– [Electronic Resource] / Natarajan S., Vairavasundaram S., Natarajan S., Gandomi A.H. // Expert Syst Appl, 2020. – Access Path: https://www.semanticscholar.org/paper/

28. Neculoiu P., Versteeghand M., Rotaru M. Learning Text Similarity with Siamese Recurrent Networks. – [Електронний ресурс] – 2016. – Access Path: https://www.researchgate.net/publication/

29.Paleti L., Krishna P.R., Murthy J. Approaching the cold-start problem using community detection based alternating least square factorization in recommendation systems. / Paleti L., Krishna P.R., Murthy J. // Evol Intell, 2020. – 14.– P. 835–49.– Access Path: https://www.researchgate.net/publication

30. Pei T., Jasra A., Hand D., Zhu A.X., Zhou C. Decode: a new method for discovering clusters of different densities in spatial data. – [Електронний ресурс] / Pei T., Jasra A., Hand D., Zhu A.X., Zhou C. // Data Mining and Knowledge Discovery, 2009. – P. 337–369. – Access Path: https://www.researchgate.net

31. Sander J. Density-based clustering in Spatial Databases. – [Електронний ресурс] / Data Mining and Knowledge Discovery, 1998. – P. 169-194. – Access Path: https://link.springer.com

32. Sato R., Makoto Yamada, Hisashi Kashima. Re-evaluating Word Mover's Distance. – [Electronic Resource] / Sato R., Makoto Yamada, Hisashi Kashima// Proceedings of the 39th International Conference on Machine Learning, PMLR, 2022. – C.19231-19249. – Access Path: https://proceedings.mlr.press/v162/

33. Stuetzle W., Nugent R. A generalized single linkage method for estimating the cluster tree of a density. – [Електронний ресурс] / Stuetzle W., Nugent R. // Journal of Computation and Graphic Statistics. – 19(2), 2010. – P. 397–418. – Access Path: https://www.researchgate.net

34. Sun B., Ma Q., Zhang S., Liu K., Liu Y. iSelf: towards cold-start emotion labeling using transfer learning with smartphones. – [Electronic Resource] /Sun B., Ma Q., Zhang S., Liu K., Liu Y.// ACM Trans Sens Netw, 2017. – 13. – P. 1–22. – Access Path: https://innovationcenter.msu.edu/wp-content

35. Sun H., Huang J., Han J., Deng H., Zhao P., Feng, B. gSkeletonClu: Density-based network clustering via structure-connected tree division or

agglomeration . – [Електронний ресурс] / Sun H., Huang J., Han J., Deng H., Zhao P., Feng, B. // Proceedings of the IEEE Int. Conf. Data Mining, 2010. – Access Path: https://ieeexplore.ieee.org

36. Tan P., Steinbach M., Kumar V. Introduction to Data Mining. – [Електронний ресурс] / Tan P., Steinbach M., Kumar V. // Pearson, 2006. – 169 p. – Access Path: https://www-users.cse.umn.edu

37. Thyagarajan A., Mueller J. Siamese Recurrent Architectures for Learning Sentence Similarity. – [Електронний ресурс] / Thyagarajan A., Mueller J. // Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2018. – Access Path: https://www.researchgate.net/publication

38.https://www.ukrinform.ua/rubric-economy/3662630-itindustria-prinesla-ekonomici-ukraini-torik-734

39.https://www.analyticsinsight.net/top-10-programming-languages-in-2023-with-the-largest-developer-communities/

40. https://hdbscan.readthedocs.io/en/latest/basic_hdbscan.html

41. https://numba.pydata.org/

42. https://scikit-learn.org/stable/user_guide.html

43. https://numpy.org/doc/stable/user/whatisnumpy.html

44. https://spacy.io/api

45. https://pandas.pydata.org/about/index.html

46. https://www.activestate.com/resources/quick-reads/what-is-matplotlib