УДК 004.023 DOi 10.30837/ bi.2021.1(96).09

В.Є. Байдак[1], **О.О. Мазурова**[2], **О.Г. Ворочек**[3]

[1]магістрант кафедри програмної інженерії ХНУРЕ, м. Харків,
vadym.baidak@nure.ua, ORCID ID 0000-0001-8166-181X

[2] кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ, м. Харків,
oksana.mazurova@nure.ua, ORCID ID 0000-0003-3715-3476

[3]кандидат технічних наук, доцент кафедри програмної інженерії ХНУРЕ, м. Харків,
olga.vorochek@nure.uamailto:oksana.mazurova@nure.ua, ORCID ID 0000-0002-9054-9894

РОЗРОБКА КОМБІНОВАНОГО МЕТОДУ ПОБУДОВИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН-МАГАЗИНУ ЕЛЕКТРОННИХ ІГОР

Запропоновано комбінований метод побудови рекомендаційної системи для розширення функціоналу онлайн-магазину електронних ігор. Запропонований метод, що враховує аналітичний профіль користувача та оцінки інших користувачів, дозволить системі надавати більш точні персоналізовані рекомендації. Основою побудови рекомендаційної системи пропонується метод колаборативної фільтрації. Для покращення якості рекомендацій пропонується вибір вхідних даних для колаборативної фільтрації на базі кластеризації користувачів за допомогою методу k-means на основі аналітичного профіля користувача. Розроблений метод апробований під час створення рекомендаційної системи для онлайн-магазину електронних ігор.

ОНЛАЙН-МАГАЗИН, РЕКОМЕНДАЦІЙНА СИСТЕМА, АНАЛІТИЧНИЙ ПРОФІЛЬ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КЛАСТЕРИЗАЦІЯ, K-MEANS

Байдак В.Е., Мазурова О.А., Ворочек О.Г. Разработка комбинированного метода построения рекомендательной системы для онлайн-магазина электронных игр. Предложен комбинированный метод построения рекомендательной системы для расширения функционала онлайн-магазина электронных игр. Предложенный метод, учитывающий аналитический профиль и оценки других пользователей, позволит системе предоставлять более точные персонализированные рекомендации. Основой построения рекомендательной системы предлагается метод колаборативной фильтрации. Для улучшения качества рекомендаций предлагается выбор входных данных для колаборативной фильтрации на базе кластеризации пользователей с помощью метода k-means на основе аналитического профиля пользователя. Разработанный метод апробирован при создании рекомендательной системы для онлайн-магазина электронных игр.

ОНЛАЙН-МАГАЗИН, РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, АНАЛИТИЧЕСКИЙ ПРОФИЛЬ, КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ, КЛАСТЕРИЗАЦИЯ, K-MEANS

Baidak Vadym, Mazurova Oksana, Vorochek Olga.Development of a combined method for constructing a recommendation system for an online store of electronic games. A combined method for constructing a recommendation system for expanding the functionality of an online store of electronic games is proposed. The proposed method, which takes into account the analytical profile of the user and the assessments of other users, will allow the system to provide more accurate personalized recommendations. The method of collaborative filtration is proposed as the basis for building a recommendation system. To improve the quality of recommendations, it is proposed to select input data for collaborative filtering based on user clustering using the k-means method based on an analytical user profile. The developed method was tested when creating a recommendation system for an online store of electronic games.

ONLINE STORE, RECOMMENDATION SYSTEM, ANALYTICAL PROFILE, COLLABORATIVE FILTRATION, CLUSTERING, K-MEANS

1. Постановка проблеми

На сьогоднішній день, великий потік інформації, доступний людині, мимоволі ініціює роботу з пошуку і фільтрації даних в ньому. Важливим інструментом отримання інформації сучасною людиною є пошукові системи. Однак при відсутності чіткого розуміння конкретного запиту складно знайти необхідну інформацію. Цю проблему частково вирішують рекомендаційні системи (РС), що здатні знаходити об'єкти схожі на те, що подобається або потрібно користувачеві на основі інформації про нього та схожих користувачів. РС аналізують інтереси користувачів і намагаються передбачити, що саме буде найцікавіше для конкретного користувача в даний момент часу [1].

РС є критично важливими в деяких галузях, адже їх ефективне використання може принести величезний дохід або дозволить суттєво виділитися серед конкурентів. РС зазвичай використовуються в сфері комерції на сайтах продажу продуктів. Під продуктом мається на увазі товар або послуга, яку можна запропонувати для ринку, і яка буде задовольняти потреби споживачів [2]. В цій комерційній сфері РС може виявити потреби відвідувачів онлайн-системи і зробити цікаві саме їм пропозиції, збільшуючи дохід системи за рахунок зростання конверсії, середнього чека і частоти повторних покупок [3].

Найважливішою якістю будь-якої РС є точність рекомендацій, які повинні бути персоналізованими і відповідати вподобанням користувачів. РС, яка не володіє такою якістю, не тільки не буде приносити ніякої користі ані користувачам, ані власникам сервісів, а навіть може нанести фінансовий збиток. Таким

чином, існує потреба в розробці методів побудови рекомендаційних систем, які дозолять їм надавати точні персоналізовані рекомендації.

2. Аналіз основних досліджень

Традиційно для побудови РС використовуються підходи на базі колаборативної фільтрації, контентної фільтрації або гібридний підхід. Колаборативна фільтрація рекомендує елементи, визначаючи інших користувачів зі схожими вподобаннями. Контентна фільтрація співвідносить властивості контенту і характеристики користувача. Гібридний підхід поєднує в собі інструменти колаборативної і контентної фільтрації [4].

Колаборативний підхід є найбільш зрілим та реалізується найпоширеніше. Вхідними даними для колаборативної фільтрації зазвичай є набір оцінок усіх користувачів сервісу і це призводить до одного з її основних недоліків, так званої проблеми «білих ворон», що полягає у погіршенні рекомендацій певним категоріям користувачів. Деякі користувачі мають специфічні уподобання, які не узгоджується з уподобаннями інших користувачів [5]. Точність рекомендацій для таких користувачів знижується і оцінки таких користувачів можуть негативно вплинути на точність рекомендацій іншим категоріям користувачів [6].

Якщо користувачів кластеризувати за визначеним аналітичним профілем і використовувати в колаборативній фільтрації для надання рекомендацій данні оцінок користувачів одного кластеру, це дозволить збільшити точність рекомендацій колаборативної фільтрації.

3. Постановка задачі

Отже, була поставлена задача розробити комбінований метод побудови РС, що використовує метод колаборативної фільтрації і забезпечує найбільш повне використання всіх даних користувачів за рахунок використання кластеризації користувачів. Практичну апробацію методу необхідно провести на прикладі побудови рекомендаційної онлайн-системи продажу комп'ютерних ігор, та за її результатами розробити рекомендацій, щодо побудови РС, що найбільш адекватно відображають очікування відвідувачів подібних систем.

4. Розробка комбінованого методу побудови РС

Запропонований комбінований метод передбачає поєднання у собі алгоритмів колаборативної фільтрації для надання рекомендацій і k-means кластеризацію [1] на основі аналітичних профілів користувачів для поліпшення рекомендацій.

Вхідними даними для кластеризації пропонується використовувати характеристики користувачів нормовані до шкали від 1 до h, на основі яких створюються кластери «схожих» користувачів за допомогою алгоритму k-means:

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & \cdots & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & \cdots & \cdots & x_{mn} \end{pmatrix} \overset{k-means}{\Rightarrow} \begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix}$$

де x_{ij} – значення j-ої характеристики для i-го користувача, n – кількість характеристик, m – кількість користувачів, c_i – кластер отриманий в результаті роботи алгоритму k-means, k – кількість кластерів.

Після побудови кластерів «схожих» користувачів, для кожного кластера вирішено застосувати алгоритм машинного навчання на основі колаборативної фільтрації (факторизація матриці з градієнтним спуском [7]) для побудови рекомендаційних моделей. Отримані моделі, у свою чергу, будуть використовуватися для прогнозування оцінок продуктів користувачами:

$$\begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix} \overset{collaboration\ filtering}{\Rightarrow} \begin{pmatrix} m_1 \\ m_2 \\ \cdots \\ m_k \end{pmatrix}$$

де m_i – модель для побудови рекомендації для i-го кластеру.

Для вибору рекомендованих продуктів з вхідного набору (продукти, які користувач ще не оцінював) буде виконуватися наступне:

– вибір моделі відповідно до кластеру користувача;

– застосування моделі до користувача і кожного продукту з вхідного набору – отримання прогнозованих оцінок продуктів користувачем;

– вибір для рекомендації тих продуктів, прогнозована оцінка яких перевищує певний поріг (зазвичай береться 3,5).

Отже, вибір рекомендованих продуктів можна представити, як:

$$\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix} + u_{c_i} \overset{m_i}{\Rightarrow} \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix} \overset{r_j > r_{threshold}}{\Rightarrow} \begin{pmatrix} p_f \\ \cdots \\ p_l \end{pmatrix}, \begin{pmatrix} p_f \\ \cdots \\ p_l \end{pmatrix} \in \begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix}$$

де $\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix}$ – вхідний набір продуктів для рекомендацій, n – кількість продуктів у вхідному наборі даних, u_{c_i} – користувач з кластеру c_i, для якого будуються рекомендації, m_i – рекомендаційна модель для кластеру c_i, $\begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix}$ – отримані прогнозовані значення оцінок продуктів користувачем, $r_{threshold}$ – поріг рекомендації

(зазвичай береться 3,5), $\begin{pmatrix} g_f \\ \cdots \\ \cdots \\ g_l \end{pmatrix}$ – продукти, обрані для рекомендації користувачу.

Кожен новий користувач повинен бути віднесений до існуючих кластерів, або кластери можуть бути перебудовані. Для нових користувачів рекомендації можуть будуватися на основі найпопулярніших продуктів в рамках його кластеру.

5. Моделювання колаборативної фільтрації для запропонованого методу

Представимо вхідний набір даних для колаборативної фільтрації у вигляді матриці *C*. Рядки матриці *C* відповідають за продукти, а стовпці за користувачів. Оцінки можуть приймати значення від 1 до 5, якщо користувач не оцінював продукт, то відповідна комірка ініціалізується як 0. Приклад матриці *C* наведений на рисунку 1.

$$\begin{array}{c} \\ \text{продукти} \end{array} \begin{array}{c} \text{користувачі} \\ \begin{bmatrix} 5 & 0 & 3 & \dots & 0 & 3 \\ 0 & 4 & 0 & \dots & 2 & 4 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 3 \\ 5 & 0 & 3 & \dots & 0 & 0 \end{bmatrix} \end{array}$$

Рис. 1. Матриця оцінок продуктів користувачами

Нехай наступний набір векторів описує *h* характеристик кожного продукту:

$$p^{(1)}, p^{(2)}, \dots, p^{(\text{кількість продуктів})}\ p^i \in \mathbb{R}^h$$

Інший набір векторів описує відношення користувача до кожної з характеристик:

$$u^{(1)}, u^{(2)}, \dots, u^{(\text{кількість користувачів})}\ u^i \in \mathbb{R}^h$$

Отже, вектори $u^{(j)}$ і $p^{(i)}$ мають однакову розмірність, а параметр $u_n^{(j)}$ відображає ставлення користувача *j* до характеристики $p_n^{(i)}$ продукту *i*. У такому випадку оцінку даного продукту цим користувачем можна розрахувати як $\left(u^{(i)}\right)^T p^i$. Надалі будемо позначати кількість користувачів як n_u, а кількість продуктів як n_p. Уявімо, що набір векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ відомий. В такому випадку можна знайти значення вектору $p^{(j)}$ за допомого мінімізації наступного функціоналу:

$$F = \frac{1}{2} \sum_{j:r(i,j)} \left(\left(u^{(j)}\right)^T p^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{l=1}^{n} \left(p_l^i\right)^2$$

де $r(i,j) = \begin{cases} 1, \text{якщо } j \text{ користувач оцінив } i \text{ продукт} \\ 0, \text{якщо не оцінив} \end{cases}$,

$\frac{\lambda}{2} \sum_{l=1}^{n} \left(p_l^i\right)^2$ – регулярізація

За допомогою цієї функції розраховується квадратична помилка того, на скільки відрізняються оцінки, які були отримані за допомогою параметрів u і p, в порівнянні зі справжніми значеннями. Отже, після мінімізації функціоналу F, ми отримаємо деякі характеристики $p^{(i)}$ для *i* продукту. Але нам потрібні характеристики для всіх продуктів, в цьому випадку функціонал буде мати наступний вигляд:

$$F\left(p^{(1)}, \dots, p^{(n_p)}\right) = \\ = \frac{1}{2} \sum_{i=1}^{n_p} \sum_{j:r(i,j)=1} \left(\left(u^{(j)}\right)^T p^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_p} \sum_{l=1}^{n} \left(p_l^i\right)^2 \quad (1)$$

Розглянемо зворотну ситуацію коли нам відомі характеристики продуктів $p^{(1)}, p^{(2)}, \dots, p^{(n_g)}$. Тоді значення векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ можна знайти за допомогою мінімізації наступного функціоналу:

$$F\left(u^{(1)}, \dots, u^{(n_u)}\right) = \\ = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(\left(u^{(j)}\right)^T p^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{l=1}^{n} \left(u_l^j\right)^2 \quad (2)$$

Однак обидва набори векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ і $p^{(1)}, p^{(2)}, \dots, p^{(n_p)}$ нам невідомі. Існує два способи вирішення даної проблеми – за допомогою алгоритму найменших квадратів, що чергуються (Alternating Least Squares) або за допомогою методу факторизації матриці низького рангу (Low Rank Matrix Factorization) [7].

Можна обчислювати обидва набори u і p одночасно. З функціоналів (1) і (2), можна виявити, що вони обчислюють однакову суму за винятком регуляризації. Перший функціонал обчислює суму помилок для користувачів, які оцінили даний продукт. Другий функціонал для кожного користувача обчислює суму квадратних відхилень для тих продуктів, які були оцінені даним користувачем. Отже, обидва функціонала обчислюють суму відхилень для всіх пар користувач-продукт, які мають значення 1 в таблиці *r*, тобто даний користувач переглянув і оцінив даний продукт. Отримаємо наступний функціонал (функція вартості або функція втрат), який потрібно мінімізувати:

$$F\left(p^{(1)}, \dots, p^{(n_p)}, u^{(1)}, \dots, u^{(n_u)}\right) = \\ = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\left(u^{(j)}\right)^T p^i - y^{(i,j)} \right)^2 + \quad (3) \\ + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{l=1}^{n} \left(u_l^j\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_p} \sum_{l=1}^{n} \left(p_l^i\right)^2$$

Вектори u і p ініціалізуються маленькими випадковими значеннями. Після того, як обчислені всі характеристики, можна отримати наступну матрицю прогнозувань:

$$\begin{bmatrix} \left(u^{(1)}\right)^T p^{(1)} & \left(u^{(2)}\right)^T p^{(1)} & \cdots & \left(u^{(n_u)}\right)^T p^{(1)} \\ \left(u^{(1)}\right)^T p^{(2)} & \left(u^{(2)}\right)^T p^{(2)} & \cdots & \left(u^{(n_u)}\right)^T p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \left(u^{(1)}\right)^T p^{(n_p)} & \left(u^{(2)}\right)^T p^{(n_p)} & \cdots & \left(u^{(n_u)}\right)^T p^{(n_p)} \end{bmatrix} \quad (4)$$

У цій матриці містяться прогнозування оцінок для всіх продуктів всіма користувачами. Наприклад, прогноз оцінки користувача *j* для гри *i* обчислюється як $\left(u^{(j)}\right)^T p^{(i)}$.

Приведемо матрицю (4) до векторизованого вигляду. Для цього введемо матриці *P* і *U*:

$$P=\begin{bmatrix} \cdots & \left(p^{(1)}\right)^T & \cdots \\ \cdots & \left(p^{(2)}\right)^T & \cdots \\ & \vdots & \\ \cdots & \left(p^{(n_p)}\right)^T & \cdots \end{bmatrix}$$

$$U=\begin{bmatrix} \cdots & \left(u^{(1)}\right)^T & \cdots \\ \cdots & \left(u^{(2)}\right)^T & \cdots \\ & \vdots & \\ \cdots & \left(u^{(n_u)}\right)^T & \cdots \end{bmatrix}$$

Матриця *P* містить характеристики усіх продуктів і має розмір n_p на *k*. Матриця *U* містить значення ставлення користувачів до характеристик продукту і має розмір n_u на *k*.

Таким чином, матрицю прогнозування можна подати наступним чином $C = P\left(U\right)^T$.

Матрицю *C* апроксимується за допомогою представлення її як добутку двох матриць. Метою цієї апроксимації є отримання передбачень для ігор, які ще не були оцінені користувачем, тобто нулі в початковій матриці повинні бути замінені на прогнозні значення. Досягти цієї мети допомагає той факт, що мінімізуємий функціонал враховує помилки тільки для тих комірок матриці, для яких вже були проставлені оцінки.

6. Оптимізація градієнтного спуску

Для обчислення наборів векторів $u^{(1)},u^{(2)},\ldots,u^{(n_u)}$ і $p^{(1)},p^{(2)},\ldots,p^{(n_p)}$ пропонується використовувати метод факторизації матриці низького рангу як більш простий в реалізації. Цей метод передбачає вирішення задачі мінімізації функції витрат (3).

Існує декілька алгоритмів призначених для вирішення задачі мінімізації, але одним з найпопулярніших є алгоритм градієнтного спуску. Суть даного алгоритму полягає в тому, що параметри функції втрат оновлюються в протилежному напрямку градієнта даної функції. Тобто, якщо при поточних параметрах функція зростає (градієнт додатний), то її мінімум знаходиться зліва, і додатний градієнт віднімається. Якщо градієнт від'ємний, то мінімум функції знаходиться праворуч і ми віднімаємо від'ємний градієнт:

$$\omega = \omega - \lambda \times \nabla_\omega F\left(\omega\right) \qquad (5)$$

де ω – параметри, $J\left(\omega\right)$ – функція втрат, λ – коефіцієнт швидкості навчання (learning rate).

З формули (5) можна побачити, що параметри оновлюються на величину кратну градієнту. Коефіцієнт швидкості навчання використовується для регуляції швидкості сходження алгоритму та його стабільності. Якщо значення коефіцієнту занадто велике, то алгоритм може розходитися, а якщо занадто маленьке, то алгоритм може сходиться дуже повільно.

Градієнт для векторів $p^{(1)},p^{(2)},\ldots,p^{(n_g)}$ при функції втрат (3) обчислюється наступним чином:

$$\frac{\partial Y}{\partial p_k^{(i)}} = \sum_{j:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)u_k^{(j)} + \lambda p_k^{(i)}$$

де λ – коефіцієнт регулярізації.

Для набору $u^{(1)},u^{(2)},\ldots,u^{(n_u)}$:

$$\frac{\partial Y}{\partial u_k^{(j)}} = \sum_{i:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)p_k^{(i)} + \lambda u_k^{(j)}$$

де λ – коефіцієнт регулярізації.

Алгоритм градієнтного спуску складається з трьох кроків:

а) задаються початкові значення для параметрів (для колаборативної фільтрації беруться випадкові маленькі значення) і точність розрахунку ε;

б) за формулою (5) обчислюються оновлені значення параметрів;

в) перевіряється умова зупинки алгоритму. Наприклад, алгоритм припиняє роботу, якщо функція втрат змінилася на величину меншу, ніж ε, інакше переходить в пункт б).

Такий алгоритм називається алгоритмом пакетного градієнту (Batch Gradient Descent). Але цей алгоритм не застосовується коли необхідно працювати з великими даними, тому що необхідно робити обчислення для всього набору даних для кожного оновлення. Однак, існує альтернативний алгоритм, який має назву алгоритм стохастичного градієнта (Stochastic Gradient Descent). В цьому алгоритмі оновлення параметрів виконується для кожної пари вхідних даних. Отже, замість формули (5) застосовується формула:

$$\omega = \omega - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right) \qquad (6)$$

де ω – параметри, $(x^{(i)}; y^{(i)})$ – приклад з вхідного набору даних.

Градієнтний спуск сходиться в локальний мінімум і це є його значним недоліком. Потрапляння в локальний мінімум може істотно погіршити якість прогнозів і оскільки функція втрат (3.3) має безліч локальних мінімумів, то цей недолік є суттєвим. Спробувати вирішити цю проблему можна намагаючись запустити алгоритм декілька разів, щоб параметри були ініціалізовані в такому місці, де можливо уникнути локальні мінімуми. Але на це може піти величезна кількість спроб, враховуючи специфіку

функції втрат, і не має гарантії, що алгоритм зійшовся саме в глобальний мінімум.

Однак, можливо уникнути проблеми локальних мінімумів, оптимізувавши класичний алгоритм градієнтного спуску. До найбільш ефективних і простих оптимізацій відноситься оптимізація, яка носить назву «Імпульс» (Momentum) [7].

Оптимізація типу імпульс потребує заміну формули (6) на формули:

$$v_t = 0.9 \times v_{t-1} - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right)$$

$$\omega = \omega - v_t$$

Імпульс для кожного параметра накопичується в векторі v. Отже, для мінімізації функції витрат (3) пропонується використовувати описаний оптимізований алгоритм градієнтного спуску.

7. Моделювання кластеризації для розбиття користувачів на групи

Для підвищення ефективності рекомендаційної системи та поліпшення якості рекомендацій пропонується використовувати кластеризацію користувачів. Характеристики користувача повинні бути виражені в чисельних показниках і представлені масивом чисел від 1 до n після нормування. Для кластеризації даних подібного типу найчастіше використовують алгоритм k-means. [8].

Даний алгоритм розділяє безліч вхідних об'єктів на задане число кластерів. Алгоритм починає свою роботу з ініціалізації центрів кластерів (центроїдів):

$$\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$$

де k – кількість кластерів, n – розмірність кожного об'єкта.

Кожен елемент множини відноситься до кластеру з найближчим центром. Наступні етапи повторюються до тих пір, поки алгоритм не зійдеться.

Далі заповнюється вектор c. Цей процес записується наступним чином – for $i = 1$ to l:

$$c^{(i)} = \left\{ p : x^{(i)} - \mu_p{}^2 \le x^{(i)} - \mu_j{}^2 \forall j, 1 \le j \le K \right\} \qquad (7)$$

де $x^{(i)}$ – об'єкт з набору вхідних даних; c – вектор, який містить відповідність між об'єктом $x^{(i)}$ і кластером $c^{(i)}$, до якого він належить; l – кількість об'єктів множини вхідних даних.

З формули (7) виходить, що кожен об'єкт з набору вхідних даних відноситься до кластеру з найближчим центром.

На наступному етапі оновлюється положення центроїдів – for $k = 1$ to K:

$$\mu_k = \frac{1}{S_k} \sum_{x^{(j)}: c^{(j)} = k} x^{(j)} \qquad (8)$$

З формули (8) виходить, що значення центру має безлічі вхідних об'єктів присвоюються кожному центру кластера k, до якого належать ці об'єкти. Якщо не відбулося зміни положення центрів кластерів після поточної ітерації, то алгоритм вважається завершеним.

Однак у k-means алгоритму є ряд недоліків:

– число кластерів визначається заздалегідь;

– не гарантується досягнення глобального мінімуму;

– може відбуватися різне розбиття для однакових вхідних даних при різних початкових положеннях центрів кластерів.

Розглянемо підходи до мінімізації даних недоліків. Введемо функцію втрат для оцінювання якості розбиття:

$$F\left(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K\right) = \frac{1}{m} \sum_{i=1}^{m} x^i - \mu_{c^{(i)}}{}^2$$

Наведений функціонал обчислює суми квадратичних відстаней від центрів кластерів до об'єктів відповідних цим кластерам. Метою алгоритму оптимізації кластеризації є мінімізація даного функціоналу.

Розглянемо перший недолік – число кластерів визначається заздалегідь. Цей недолік можна мінімізувати, використавши метод ліктя. За цим методом розробник самостійно обирає кількість кластерів і оцінює зменшення функції втрат, при збільшенні кількості кластерів.

Другий недолік полягає у відсутності гарантії досягнення глобального мінімуму. Даний недолік можливо мінімізувати за допомогою множинної ініціалізації. Алгоритм k-means ініціалізується різними наборами центроїдів, для кожного набору оцінюється значення функції витрат – обирається той варіант, при якому функція втрат має найменше значення.

8. Використання комбінованого методу побудови рекомендаційної системи в онлайн-магазині електронних ігор

За допомогою розробленого комбінованого методу була побудована рекомендаційна система для онлайн-магазину електронних ігор.

Для аналітичного профілю користувача, на основі якого відбувається кластеризація, були обрані наступні характеристики – вік, стать, улюблені жанри ігор. Для жанру були виділені кількісні характеристики, які можна перенести на користувача, у якого цей жанр є улюбленим. До цих характеристик були віднесені – рівні навичок, екшену, стратегічності, сюжету, зрілості, насилля та освіти. На рис. 2 наведено схему бази даних для онлайн-магазину електронних ігор. Для роботи РС потрібні такі сутності бази даних, як гра, жанр, користувач, оцінка гри користувачем.

Вхідні дані (характеристики користувачів та ігор, оцінки ігор користувачами) для розробленого комбінованого методу були отримані за допомогою API найкрупнішої онлайн-системи продажу ігор – Steam.

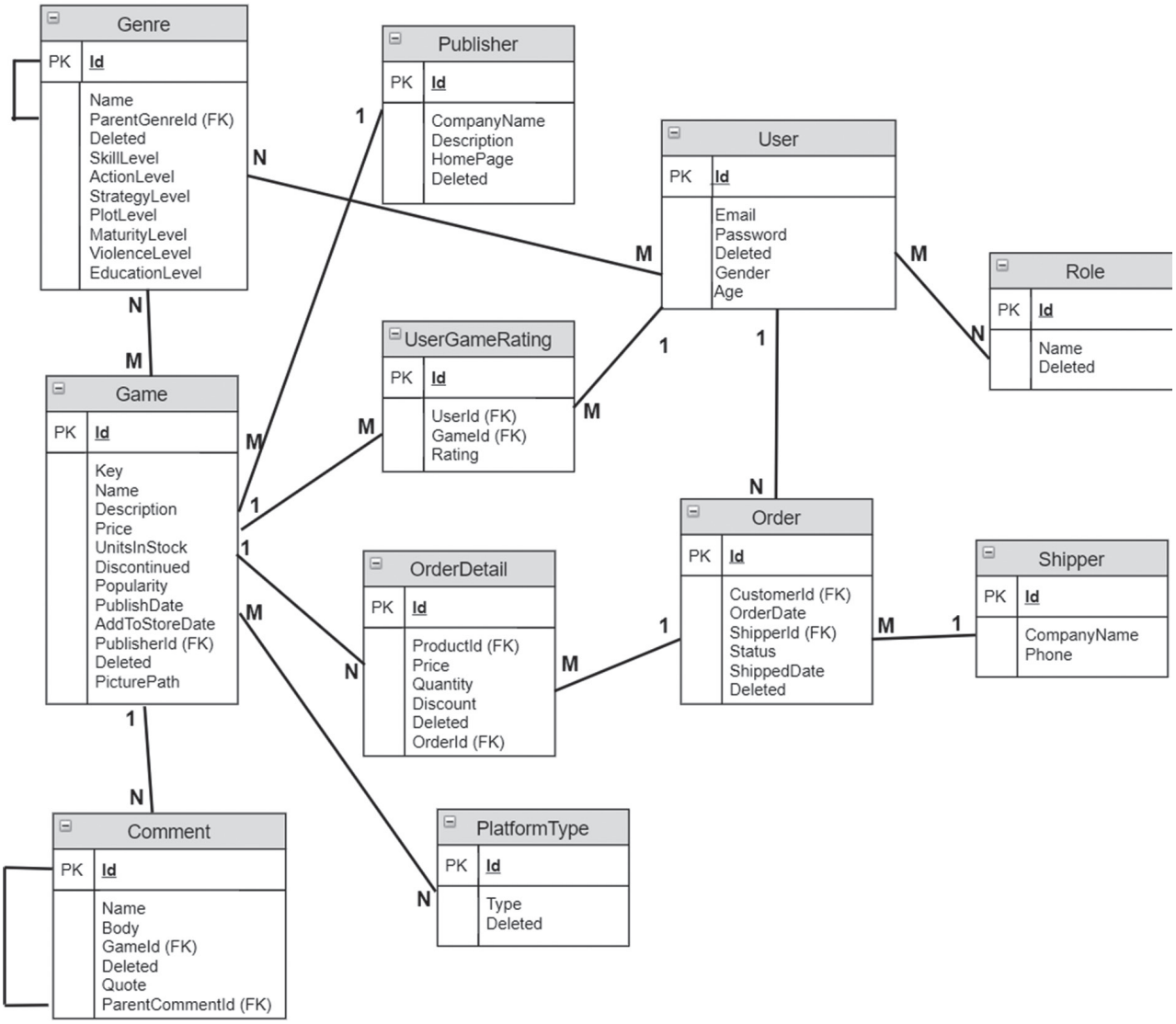

Рис. 2. Схема бази даних онлайн-магазину електронних ігор

Оскільки Steam API не надає інформацію про оцінки ігор користувачами, значення кількості зіграних годин були використані для конвертації у оцінку.

Для оцінки ефективності розробленої на базі запропонованого методу РС та точності рекомендацій РС використовувалася метрика RMSE (Root of Mean Squared Error) [6]. RMSE використовується для вимірювання різниці між передбачуваними значеннями моделі РС та спостережуваними значеннями тестового набору даних. Чим цей показник нижче, тим ефективніша модель і точніші рекомендації.

Для вибору кількості кластерів для розробленої РС була проведена низка експериментів з замірюванням RMSE показника та показника R-квадрат. Показник R-квадрат вказує наскільки дані відповідають моделі. Значення цього показника 0 означає, що дані є випадковими та не можуть відповідати моделі. Значення 1 — що модель точно відповідає даним. Показники RMSE та R-квадрат для різної кількості кластерів наведені у табл. 1.

При кількості кластерів більше 6, показник R-квадрат все більше зменшувався, тому в табл. 1 наведені результати експериментів тільки для кластерів від 2 до 6. Для розробленої РС було обрано 5 кластерів для розбиття, оскільки при цій кількості кластерів РС показує найменший показник RMSE та найбільший показник R-квадрат. Також було проведене порівняння показника RMSE РС, яка використовує колаборативну фільтрацію, та розробленої РС, яка використовує колаборативну фільтрацію на основі методу факторизації матриці низького рангу та k-means кластеризацію для розбиття користувачів на групи за аналітичним профілем. В табл. 2 наведене порівняння показника RMSE для наведених РС для користувачів з різною кількістю оцінок.

Таблиця 1

Показники	Кількість кластерів				
	2	3	4	5	6
RMSE	0.96	0.96	1.11	0.78	1.01
R-квадрат	0.54	0.53	0.48	0.72	0.32

Таблиця 2

RMSE для РС	Користувачі (кількість оцінок)				
	16	42	84	112	166
з *k*-means кластеризацією та колаборитивною фільтрацією	0.82	0.88	0.78	0.93	0.99
з колаборитивною фільтрацією	1.53				

RMSE для РС з колаборативною фільтрацією має одне і теж значення для різних користувачів оскільки використовується одна і та сама рекомендаційна модель для всіх користувачів. У випадку РС з k-means кластеризацією, рекомендаційна модель відрізняється в залежності від того до якого кластеру був віднесений користувач. Як можна побачити з табл. 2, показники RMSE для РС з k-means кластеризацією нижчі (в деяких випадках майже в 2 рази) ніж для РС, яка використовує тільки колаборативну фільтрацію. Отже розроблена система будує більш ефективні рекомендаційні моделі і робить точніші рекомендації.

Розроблена рекомендаційна система надає точні персоналізовані рекомендації ігор на основі оцінок цільового користувача та інших користувачів системи з використанням кластеризації користувачів для підвищення точності рекомендацій для користувачів в рамках кластеру.

Розроблена РС реалізовує наступні функції:

– додавання користувачем оцінок та відгуків для ігор;

– створення профілю користувача з інформацією про нього;

– додавання користувачем інформації про улюблений жанр, платформу, видавця;

– кластеризація користувачів на основі профілів користувачів за допомогою методу k-means;

– тренування моделі колаборативної фільтрації для окремих кластерів користувачів;

Розроблена система базується на фреймворці для створення веб-додатків ASP .NET MVC 5 та на бібліотеці машинного навчання ML.NET. Для доступу до реляційних даних використовується СУБД MS SQL Server 2016, ORM Entity Framework та FluentAPI. Інтерфейс програмної системи дозволяє зручно оцінювати ігри, переглядати загальні середні оцінки ігор, історію оцінок та персоналізовані рекомендації.

9. Висновки та перспективи

Розроблений комбінований метод побудови рекомендаційної системи дозволяє підвищити точність персоналізованих рекомендацій (зменшити RMSE показник рекомендаційних моделей) та був покладений в основу рекомендаційної системи онлайн-магазину електронних ігор. Поліпшення рекомендацій зумовлено використання кластеризації користувачів в запропонованому комбінованому методі. Розроблена рекомендаційна система, завдяки поліпшеній якості рекомендацій, буде корисна як користувачам онлайн-магазину електронних ігор, оскільки буде надавати їм релевантні пропозиції і зменшувати час на пошук бажаного, так і власникам онлайн-магазину, оскільки вона може збільшити дохід за рахунок ефективності.

Список літератури:

[1] Isinkaye F.O., Folajimi Y.O., Ojokoh B.A., Recommendation systems: Principles, methods and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2015, Pages 261-273

[2] Chalyi S., Leshchynskyi V Temporal modeling of user preferences in recommender system Chalyi S., Leshchynskyi V. CEUR Workshop Proceedings, 2020, 2711, с. 518-528

[3] Блинков Н. Объем рынка компьютерных игр URL: http://www.it-weekly.ru/market/business/73058.html (дата звернення: 25.01.2021)

[4] Doo Hyodan, Audrey Germain, Geordan Jove Recommendation System for Steam Game Store: An overview of recommender systems URL: https://audreygermain.github.io/Game-Recommendation-System/ (дата звернення: 29.01.2021)

[5] Serhii Chalyi, Volodymyr Leshchynskyi. Method of constructing explanations for recommender systems based on the temporal dynamics of user preferences. «EUREKA: Physics and Engineering».-2020.- Number 3.-p.43-50.

[6] Jena K. C., Mishra S., Sahoo S. and Mishra B. K., Principles, techniques and evaluation of recommendation systems, International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2017, pp. 1-6

[7] Yuan Lu, Jie Yang, Notes on Low-rank Matrix Factorization, URL: https://yangjiera.github.io/pdf/low-rank.pdf (дата звернення: 12.02.2021)

[8] Madushan Dileka Introduction to K-means Clustering URL: https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00 (дата звернення: 12.02.2021)

Надійшла до редколегії 11.03.2021