ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА ТЕОРІЯ КОДУВАННЯ

УДК 004.42

М. С. Даниленко, І. С. Колесник

МЕТОДИ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Вінницький національний технічний університет, Вінниця

Анотація. Розглянуто основні принципи побудови рекомендаційної системи та методи вирішення проблеми холодного старту, яка виникає внаслідок недостатньої взаємодії користувача з програмним засобом на початкових етапах роботи з ним. Збільшено ефективність роботи рекомендаційної системи за умов недостатньої вибірки даних і при появі в системі нових елементів, для яких відсутня статистика.

Ключові слова: рекомендаційна система, проблема холодного старту, веб-сервіс, машинне навчання, алгоритми.

Аннотация. Рассмотрены основные принципы построения рекомендательной системы и методы решения проблемы холодного старта, возникающей вследствие недостаточного взаимодействия пользователя с программным средством на начальных этапах работы с ним. Увеличена эффективность работы рекомендательной системы при недостаточной выборке данных и при появлении в системе новых элементов, для которых отсутствует статистика.

Ключевые слова: рекомендательная система, проблема холодного старта, веб-сервис, машинное обучение, алгоритмы.

Abstract. The basic principles of building a recommendation system and methods for solving the problem of cold start arising from insufficient interaction of the user with the software at the initial stages of working with it are considered. The efficiency of the recommender system has been increased when there is insufficient data sampling and when new elements appear in the system for which there are no statistics.

Key words: recommendation system, cold start problem, web service, machine learning, algorithms.
DOI: https://doi.org/10.31649/1999-9941-2021-52-3-10-15.

Вступ

Рекомендаційні системи стали невід'ємною частиною сучасних веб-сервісів, орієнтованих на роботу з користувачами та споживачами. Рекомендаційна система - комплекс алгоритмів, програм та сервісів, завдання якого передбачити, що може зацікавити того чи іншого користувача. В основі роботи лежить персоналізація. Вони направлені як на збільшення ефективності роботи сервісів, так і на покращення користувацького досвіду та збільшення проінформованості клієнта про надавані послуги. В залежності від моделі роботи сервісу рекомендації можуть бути як додатковою частиною до основного функціоналу, так і слугувати основою для моделі.

Актуальність

Підвищення ефективності роботи рекомендаційних систем дозволить краще персоналізовувати інформацію для користувача. Вирішення проблеми «холодного старту» покращить користувацький досвід для нових користувачів і при додаванні в систему нових сутностей.

Мета

Розглянути способи розробки та алгоритми для систем рекомендації, які є ефективними в умовах недостатньої статистичної інформації.

Задачі

1. Розглянути способи розробки та виділити основні критерії, якими визначається рекомендаційна система.
2. Розглянути архітектурні та програмні засоби для розробки.
3. Розглянути алгоритми, які є ефективними для розробки рекомендаційних систем.

Розв'язання задач

Існує чотири основних підходи до побудови прогнозів в рекомендаційних системах: колаборативна фільтрація, заснована на контенті, заснована на знаннях та гібридні.

Колаборативна фільтрація (рис. 1) − метод, який лежить в основі деяких рекомендаційних сервісів. Колаборативна фільтрація має два значення: конкретне і більш загальне. В загальному, колаборативна фільтрація − процес фільтрації інформації за допомогою засобів за участю співпраці між певною кількістю агентів, точок зору, джерел даних і т. д. Застосування колаборативної фільтрації найчастіше пов'язане з дуже об'ємними вибірками даних. Колаборативні методи фільтрації були застосовані до різних видів даних, зокрема до таких як зондування та моніторинг даних, які виникають при розвідці корисних копалин на великих площах; до фінансових даних, таких як установи фінансових послуг, які об'єднують багато фінансових джерел; або в електронній торгівлі та веб-додатках, що зосереджуються на даних користувача, і т. д. Решта цієї проблеми зосереджена на колаборативній фільтрації даних, призначених для користувача, хоча деякі з методів та підходів можуть застосовуватися так само і у багатьох інших випадках [1].

Рисунок 1 – Принцип роботи кореляційної фільтрації

У більш сучасному, вужчому значенні, колаборативна фільтрація − це один з методів побудови прогнозу в рекомендаційних системах, який використовує відомі уподобання (оцінки) групи користувачів для прогнозування невідомих уподобань іншого користувача. Основне припущення колаборативної фільтрації полягає в наступному: ті, хто однаково оцінювали будь-які предмети в минулому, схильні давати схожі оцінки інших предметів і в майбутньому. Наприклад, за допомогою колаборативної фільтрації музичний додаток здатний прогнозувати, яка музика сподобається користувачеві, маючи неповний список його уподобань (симпатій та антипатій). Прогнози складаються індивідуально для кожного користувача, хоча інформація, що використовується, зібрана від багатьох учасників. Це відрізняє колаборативну фільтрацію від більш простого підходу, дає усереднену оцінку для кожного об'єкта інтересу, наприклад того, що базується на кількості поданих за нього голосів. Дослідження в даній області активно ведуться і в наш час, що зокрема обумовлюється наявністю невирішених проблем у методі колаборативної фільтрації [1, 2].

Заснована на контенті фільтрація є основою багатьох рекомендаційних систем. На відміну від колаборативної фільтрації етап знайомства з користувачем опускається. Товари та послуги рекомендуються з урахуванням знання них: жанр, виробник, конкретні функції тощо. Загалом застосовують будь-які дані, які можна зібрати.

За таким принципом працюють системи інтернет-магазинів, онлайн кінотеатрів та інших сервісів. Наприклад, IVI вибудовує рекомендації щодо жанрів, країн-виробників фільмів, акторів тощо.

Автори платформ використовують цей тип систем, щоб не втратити нових користувачів, даних про яких ще немає. Звідси випливають два недоліки: спочатку системи діють неточно і потрібно більше часу на реалізацію [2].

Заснована на знаннях фільтрація працює на основі знань про якусь предметну область: про користувачів, товари та інші, які можуть допомогти в ранжируванні. Як і у випадку з фільтрацією за контентом, оцінки інших користувачів системи не враховують. Є кілька різновидів: case-based, demographic-based, utility-based, critique-based, whatever-you-want-based і т.д.

При реалізації нового проекту в залежності від сфери діяльності у рекомендаційну систему можна закласти будь-яку предметну область та ранжувати за нею.

Мінус фільтрації за знаннями − для розробки цієї системи потрібно багато часу та ресурсів, але результат їх виправдовує [2].

Комбінування кількох алгоритмів в рамках гібридної системи у межах однієї платформи дозволяє мінімізувати недоліки кожного. Великі сервіси та інтернет-магазини найчастіше використовують гібридні варіанти.

Існує кілька найбільш розповсюджених типів комбінування:

- реалізація окремо колаборативних та контентних алгоритмів та поєднання їх припущень;
- включення деяких контентних правил до колаборативної методики;
- включення деяких колаборативних правил до контентної методики;
- побудова загальної моделі, що включає правила обох методик.

Холодний старт − потенційна проблема в комп'ютерній інформації системи, яка включає ступінь автоматизованого моделювання даних. Зокрема, це стосується проблеми, що полягає в тому, що система не може зробити жодних висновків для користувачів або елементів, про які вона ще не зібрала достатню інформацію [3].

При проектуванні рекомендаційної системи необхідно визначити найбільш ефективний підхід для задоволення потреб сервісу, опираючись на вже наявну базу інформації, з якою може працювати система, джерела цієї інформації, орієнтованість рекомендаційної системи та можливість виникнення проблеми холодного старту або повторюваного холодного старту за умови додавання в сервіс нової інформації.

Завдання рекомендаційної системи – проінформувати користувача про товар, який може бути найцікавіший у час. Клієнт отримує інформацію, а сервіс заробляє на наданні якісних послуг. Послуги − це не обов'язково прямий продаж запропонованого товару. Сервіс також може заробляти на комісійних або просто збільшувати лояльність користувачів, яка потім виливається у рекламні та інші прибутки [3].

Залежно від моделі бізнесу рекомендації можуть бути його основою, як, наприклад, у TripAdvisor, а можуть бути просто зручним додатковим сервісом (як, наприклад, у якомусь інтернет-магазині одягу), покликаним покращити Customer Experience та зробити навігацію за каталогом. зручною.

Персоналізація онлайн-маркетингу – очевидний тренд останнього десятиліття. За оцінками McKinsey, 35% виручки Amazon або 75% Netflix припадає саме на рекомендовані товари і відсоток цей, ймовірно, зростатиме. Рекомендаційні системи – це про те, що запропонувати клієнту зробити його щасливим.

Стандартну модель рекомендаційної системи можна описати наступними параметрами:

- предмет рекомендації – що рекомендується;
- ціль рекомендації – навіщо рекомендується (наприклад: купівля, інформування, навчання, контакти);
- контекст рекомендації – що користувач робить у цей момент (наприклад: дивиться товари, слухає музику, спілкується з людьми);
- джерело рекомендації – хто рекомендує: аудиторія (середній рейтинг ресторану в TripAdvisor), схожі за інтересами користувачі, експертна спільнота (буває, коли йдеться про складний товар, такий, як, наприклад, вино);
- ступінь персоналізації. Неперсональні рекомендації – коли вам рекомендують те саме, що й іншим. Вони допускають націлення регіону або часу, але не враховують ваші особисті переваги. Більше сучасний варіант – коли рекомендації використовують дані з вашої поточної сесії. Ви переглянули кілька товарів, і внизу сторінки вам пропонуються схожі. Персональні рекомендації використовують усю доступну інформацію про клієнта, зокрема історію його покупок;
- прозорість. Люди більше довіряють рекомендації, якщо розуміють, як саме її було отримано. Так менший ризик нарватися на «несумлінні» системи, які просувають проплачений товар або дорожчі товари, що ставлять вище в рейтингу. Крім того, хороша рекомендаційна система сама повинна вміти боротися з купленими відгуками та накрутками продавців. Маніпуляції до речі бувають і ненавмисними;
- формат рекомендації. Це може бути спливаюче віконце, що з'являється в певному розділі сайту відсортований список, стрічка внизу екрана або ще щось;
- алгоритми. До найкласичніших відносяться алгоритми Summary-based (неперсональні), Content-based (моделі засновані на описі товару), Collaborative Filtering (колаборативна фільтрація), Matrix Factorization (методи засновані на матричному розкладанні) та деякі інші [3].

Для побудови рекомендаційних систем використовуються ті ж архітектурні рішення, що і для інших сервісів: монолітна архітектура, мікросервісна архітектура та сервіс-орієнтована архітектура. Монолітна архітектура забезпечує уніфікованість додатку та спрощує його розробку та розгортання для відносно невеликих проектів. При побудові системи для продажу з вбудованою рекомендаційною системою монолітна архітектура може спричинити порушення модульності та масштабованості додатку, що ускладнить його подальшу розробку і підтримку. Монолітна архітектура не дозволить швидко та якісно виконати заміну рекомендаційної системи на аналогічну або провести масштабну модернізацію сервісу.

Для програмних рішень з використанням рекомендаційних систем використання сервіс-орієнтованого архітектурного підходу (рис. 2) надасть наступні переваги:

Модульність – програмні компоненти, зокрема власне рекомендаційна система, можуть бути замінені на аналогічні або бути масштабно модернізованими;

Масштабованість – будь яка частина сервісу може бути розширена новим функціоналом, при цьому не впливаючи на коректну роботу інших сервісів;

Незалежність стеку технологій компонентів;

Надійний канал зв'язку у вигляді HTTP-протоколу [4].

Для більш комплексних рішень доцільним є використання мікросервісної архітектури.

Оскільки в рекомендаційних системах написання складних алгоритмів є більш пріоритетним, ніж швидкодія, написання додатку виконується високорівневою мовою програмування. Для написання сервісів найчастіше використовуються C# (ASP.NET), Java (Spring Framework), Python (Django) та JavaScript (Node.js).

ASP.NET − технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. На цей час останньою версією цієї технології є ASP.NET Core 6.0. ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дає змогу розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою [5].

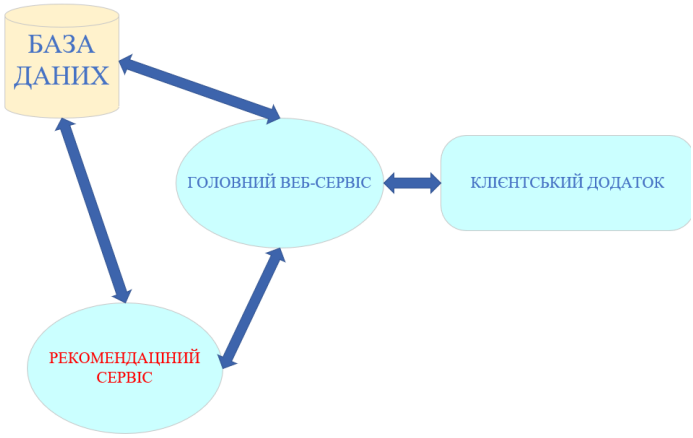

Рисунок 2 – Візуалізація використання сервіс-орієнтованої архітектури для побудови додатку з рекомендаційною системою

Spring Framework − це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB) [5].

Django (Джанго) − високорівневий відкритий Python-фреймворк (програмний каркас) для розробки веб-систем. Сайт на Django будується з однієї або декількох частин, які рекомендується робити модульними. Це одна з істотних архітектурних відмінностей цього фреймворку від деяких інших (наприклад Ruby on Rails). Архітектура Django подібна на «Модель-Вигляд-Контролер» (MVC). Однак, те що називається «контролером» в класичній моделі MVC, в Django називається «вигляд» (англ. view), а те, що мало б бути «виглядом», називається «шаблон» (англ. template). Таким чином, MVC розробники Django називають MTV («Модель-Шаблон-Вигляд») [5].

Node.js − платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Node.js має наступні властивості: асинхронна одно-нитева модель виконання запитів, неблокуючий ввід/вивід, система модулів CommonJS, рушій JavaScript Google V8. Для керування модулями використовується пакетний менеджер npm (node package manager) [5].

Проблема холодного старту є одним із основних завдань, що виникають при побудові рекомендаційних систем: що рекомендувати новим користувачам та кому показувати нові об'єкти. У безконтекстних алгоритмах це завдання вирішується за допомогою накопиченої статистичної інформації. Для використання в рекомендаційних системах розглянемо так звані «алгоритми багаторуких бандитів» та їх ефективність в умовах холодного старту [6].

Уявимо, що маємо N різних ігрових автоматів. При натисканні на ручку ми отримуємо якийсь виграш. Необхідно максимізувати сумарний прибуток всіх натискань на ручки. Завдання полягає у знаходженні оптимального способу вибору ручки на черговому кроці. Математично це можна записати так:

A – безліч доступних дій («ручок»);

Xt ∈ Rd – контекст (інформація про об'єкти та/або користувачів) на певному кроці. Він визначається середовищем, у якому розглядаються багаторукі бандити;

Pt, a, x – очікувані виплати для ручки a, які можна отримати в момент часу t при заданому контексті Xt;

Rt – реальний виграш.

$$a_t = argmax\ p_{t,a,x}, \text{де } a \in A$$
$$R_t = \sum_t r_t \to max, \text{при } t = 1, 2, \ldots$$

Розглянемо деякі види алгоритмів багаторуких бандитів, а саме UCB1 та ε-greedy, disjoint-linUCB та hybrid-linUCB. Перевірку їх ефективності проведемо на прикладі завдання Yahoo! за рекомендацією новин користувачам. При цьому використаємо метрика CTR (відношення числа кліків до показів). Результати наведено на рисунках 3, 4. При цьому під час роботи алгоритмів вважалося, що спочатку всі об'єкти та користувачі є новими для системи, оскільки до старту алгоритмів не використовувалась інформація

про історію показів. Вся робота методів заснована на статистичних даних та ознакових уявленнях об'єктів та користувачів (тільки для контекстних алгоритмів).

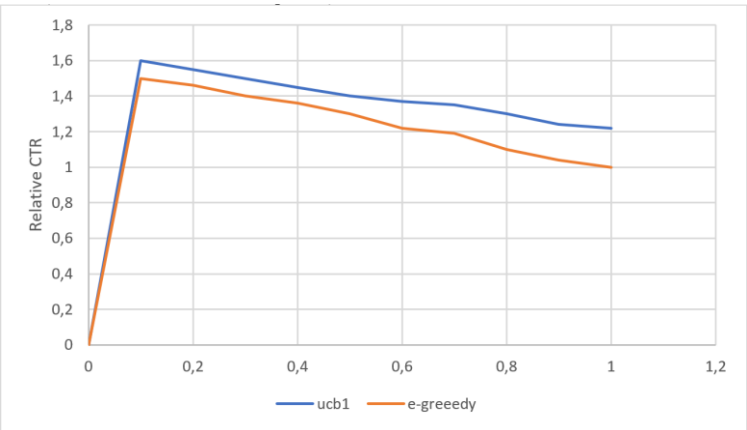

Рисунок 3 – Графік ефективності безконтекстних алгоритмів

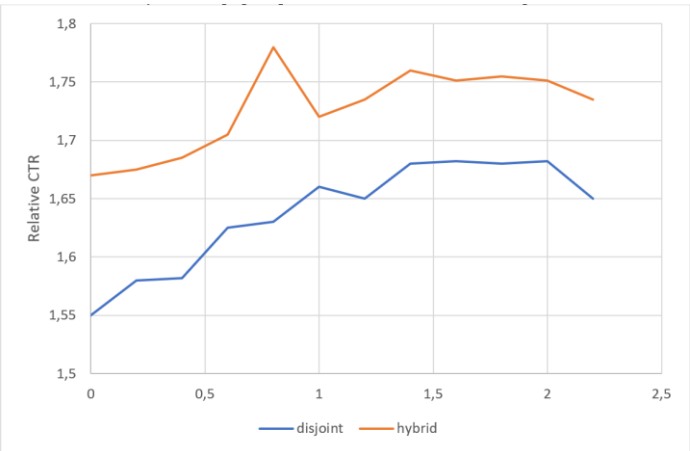

Рисунок 4 – Графік ефективності контекстних алгоритмів

Виявлено, що «бандитські алгоритми» досить добре справляються із проблемою холодного старту, що демонструють наведені графіки. Також варто зауважити, що ці методи працюють добре і для старих об'єктів.

Висновки

1. Розглянуто підхід до створення рекомендаційних систем, які є ефективними та працюють максимально ефективно в умовах малої вибірки. Були вивчені основні критерії та параметри, від яких залежить робота сервісу.

2. За рахунок використання сервіс-орієнтованої архітектури такі додатки є модульними та легко масштабованими, що надає можливість використовувати їх максимально продуктивно. В ході роботи було досліджено «бандитські алгоритми», їх ефективність та доцільність використання.

3. На основі дослідження можна зрозуміти, що вони дозволяють системі працювати в умовах холодного старту.

Список літератури

[1] К. Фальк, *Рекомендательные системы на практике*, 2020, 448 с.

[2] Li L., Chu W., Langford J., Schapire R. E., «A contextual-bandit approach to personalized news article recommendation», *Proceedings of the 19th International Conference on World Wide Web.* p. 661–670. 2010.

[3] D. Bugaychenko, A. Dzuba, «Musical recommendations and personalization in a social network», *RecSys '13: Proceedings of the 7th ACM conference on Recommender systems*, October 2013, p. 367–370.

[4] Auer P., Cesa-Bianchi N., Fischer P., «Finite-time analysis of the multiarmed bandit problem», *Machine Learning.*, vol. 47, no 2–3, p. 235–256. 2002.

[5] Yahoo! Front page today module user click log dataset, version 1.0 (1.1 GB) [Електронний ресурс]. Режим доступу: https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=49. Дата звернення: 15.11.2021.
[6] Р. З. Омаров, А. В. Востротіна, А. Д. Лі, «Проблема "холодного старту"», *Молодий учений*, № 26 (264), с. 85-88. 2019.

Стаття надійшла: 20.11.2021.

References

[1] K. Falk, *Practical recommender systems*, 2020, 448 s. [in Russian].
[2] Li L., Chu W., Langford J., Schapire R. E., «A contextual-bandit approach to personalized news article recommendation», *Proceedings of the 19th International Conference on World Wide Web.* p. 661–670. 2010.
[3] D. Bugaychenko, A. Dzuba, «Musical recommendations and personalization in a social network», *RecSys '13: Proceedings of the 7th ACM conference on Recommender systems*, October 2013, p. 367–370.
[4] Auer P., Cesa-Bianchi N., Fischer P., «Finite-time analysis of the multiarmed bandit problem», *Machine Learning.*, vol. 47, no 2–3, p. 235–256. 2002.
[5] Yahoo! Front page today module user click log dataset, version 1.0 (1.1 GB). [Online]. Available: https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=49. Accessed Nov. 15, 2021.
[6] R. Z. Omarov, A. V. Vostrotina, A. D. Li, «Problema "kholodnoho startu"», *Molodyi uchenyi*, № 26 (264), s. 85-88. 2019 [in Russian].

Відомості про авторів

Даниленко Максим Сергійович – студент групи 2КІ-20м, кафедра обчислювальної техніки.

Колесник Ірина Сергіївна – кандидат технічних наук, доцент, доцент кафедри обчислювальної техніки.

М. С. Даниленко, И. С. Колесник

МЕТОДЫ РАЗРАБОТКИ РЕКОМЕНДАЦИОННЫХ СИСТЕМ

Винницкий национальный технический университет, Винница

M. S. Danylenko, I. S. Kolesnyk

METHODS FOR DEVELOPING RECOMMENDATION SYSTEMS

Vinnytsia National Technical University, Vinnytsia