МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія рекомендації товарів для платформи онлайн продажів»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Кузіков Б.О.**

Студентка групи ІН.м – 82 **Донцова О.Є.**

СУМИ 2019

Сумський державний університет

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Донцовій Олені Євгенівні

1. Тема проекту (роботи) *Інформаційна технологія рекомендації товарів для платформи онлайн продажів*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи)
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Постановка задачі дослідження. 2) Методи розв'язання задачі. 3) Програмна реалізація відмовостійкого протоколу адміністрування розподілених систем.*

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз проблеми.		
2.	Аналіз існуючих рішень.		
3.	Порівняння гібридних підходів.		
4.	Програмна імплементація рекомендаційної системи.		
5.	Оформлення пояснювальної записки до дипломної роботи		

Студент – дипломник _____

Керівник проекту _____

РЕФЕРАТ

Записка: 73 стор., 24 рис., 2 додатки, 28 джерел.

Об'єкт дослідження — Інформаційна технологія рекомендації товарів для платформи онлайн продажів

Мета роботи — створення рекомендаційної системи для веб додатку, який є платформою онлайн продажів товарів за допомогою використання «неявних» відгуків для покращення досвіду користувача.

Результати — розроблено декілька мікросервісів, що взаємодіють між собою для зберігання даних про поведінку користувача на сайті та для обробки цих даних з метою отримання списку релевантних рекомендацій для певної особи.

РЕКОМЕНДАЦІЙНА СИСТЕМА, РЕКОМЕНДАЦІЇ, PYTHON, JAVA, APACHE, HDFS, SPARK, MAHOUT

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

РС	Рекомендаційна система
ГРС	Гібридна рекомендаційна система
CF	Алгоритм колаборативного фільтрування (Collaborative filtering)
CBF	Алгоритм фільтрування за контентом (Content based filtering)
CCO	Алгоритм колаборативного фільтрування для багатьох індикаторів, що враховує кореляцію спільності входження (Correlated Cross-Occurrence)
LLR	Алгоритм статичної перевірки відношенням правдоподібностей (Log Likelihood Ratio)

ВСТУП

Кожного дня людство створює щось нове. Близько 90% усієї наявної інформації було створено за 2 останні роки. Наші винаходи розширили фізичні можливості та збільшили свободу в просторі та часі. Кожну хвилину у світі відбувається 154.200 дзвінків у Skype, надсилається 16 мільйонів текстових повідомлень та 103.447.520 спаму у вигляді email[7].

За останнє десятиліття став більш зв’язаним та оптимізованим. Сьогодні сфера будь-якого комерційного бізнесу використовує Інтернет як джерело зручної інформації для клієнтів, для комунікаційних та рекламних цілей. Але компаніям складно самостійно оцінити нові можливості мережі. Тенденція збільшення кількості користувачів та продуктів у сфері електронної комерції, полегшення масштабування ринків, все ще має низку проблем. Серед них такі:

- пошук найкращого кандидата продукту для користувача, який найбільш точно відповідає його потребі;
- збільшення конверсії продажів продукції;
- поліпшення досвіду користування послугою електронної комерції, заохочуючи клієнта використовувати сервіс у майбутньому;

Сучасні умови економічної конкуренції вимагають від підприємців створення все нових і нових технологій приваблення покупців. Однією з таких технологій є рекомендаційні системи. Дані системи принесли нові способи взаємодії звичайних веб-сайтів зі своїми користувачами. На заміну надання статичної інформації, коли потенційні покупці шукають і, ймовірно, купують товари, рекомендаційні системи збільшують ступінь інтерактивності, а також розширюють надані користувачу можливості. Рекомендаційні системи формують рекомендації незалежно для кожного конкретного користувача на основі його минулих покупок і пошуків, а також на основі запитів інших користувачів. Також такі рекомендації збільшують чек клієнтів, завдяки пропозиції суміжних товарів або товарів, які якось пов'язані з його минулими

придбаннями. Існує безліч принципів проектування рекомендаційних систем, деякі з них будуть розібрані в даній роботі.

Завдання дослідження:

- Порівняти існуючі підходи до формування рекомендацій
- Визначити дані які є найбільш репрезентативними показниками вподобань користувача
- Проаналізувати способи відображення рекомендацій
- Спроектувати систему мікросервісів, які будуть зберігати дані та надавати рекомендації
- Розробити РС, що буде надавати рекомендації для користувача базуючись на його «неявних» відгуках – переглядах, додавання в корзину, та ін.

Предметом дослідження є програмні рішення для систем рекомендацій для електронної комерції та набір їх функціональних можливостей, які забезпечують ефективне вирішення проблеми відповідності продукту та користувача.

У галузі електронної комерції рекомендаційні системи ґрунтуються на використанні даних про поведінку користувачів у системі. У більшості випадків явних відгуків користувачів немає. Системи збирають інформацію про те, які продукти переглядав користувач, які товари були поміщені в кошик та які товари користувач придбав. Це все є неявні відгуки, оскільки користувач не виконує ніяких додаткових дій у системі, за винятком того, що він зацікавлений. Цей вид даних називається неявним відгуком, а ці огляди поділяються на багато типів: покупка товару, перегляд інформації про товар тощо. Більшість реалізацій РС не використовують усі доступні типи неявних зворотних зв’язків або не використовують найбільшу точність, втрачаючи якість рекомендацій.

Метою даної роботи є впровадження багатофакторної масштабованої системи рекомендацій для електронної комерції, яка працює на неявних даних відповідей і здатна надавати релевантні рекомендації користувачам.

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ПОРІВНЯННЯ ПІДХОДІВ ДО РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Визначення рекомендаційної системи

Фільтрування інформації - це процес моніторингу великих обсягів динамічно генерованих даних та виявлення лише того набору цих даних для користувача, який найбільш точно відповідає його потребам.

Інтерфейси систем фільтрування інформації розділяють на: фільтрувальні (filtering interfaces) та рекомендаційні (recommendation interfaces). Прикладом інтерфейсу фільтрування є фільтри електронної пошти, які розподіляють вхідні електронні листи в різні групи за конкретними атрибутами, що дозволяє користувачеві легко знаходити важливі електронні листи. Прикладом систем з рекомендаційним інтерфейсом є системи з вихідним інтерфейсом списків топ-N та списків продуктів. Елементи в цьому списку відсортовані за зменшенням коефіцієнта рекомендації товару[1].

Рекомендаційна система (РС) – це програмне забезпечення, яке, аналізуючи дані користувача, пропонує йому персональний продукт, який, ймовірно, покриває його потреби. Основною функцією РС є фільтрація певної підмножини інформації [2].

З іншого боку, завдання системи рекомендацій - це завдання сфери машинного навчання. У машинному навчанні більшість завдань передбачає навчання функції, яка буде обробляти дані. Для створення такої функції необхідний набір аргументів і значення функції від цих аргументів. В ідеалі, чим більший набір даних, тим точнішою повинна бути функція, але на практиці ми стикаємося з такими проблемами, як шум даних та неправильно підібраний набір аргументів, що спричиняє проблеми перетренування (overfitting) та недотренування (underfitting). Ці проблеми до кінця не вирішені в рекомендаційних системах. На рисунку. 1.1 можна побачити принцип функціонування рекомендаційної системи.

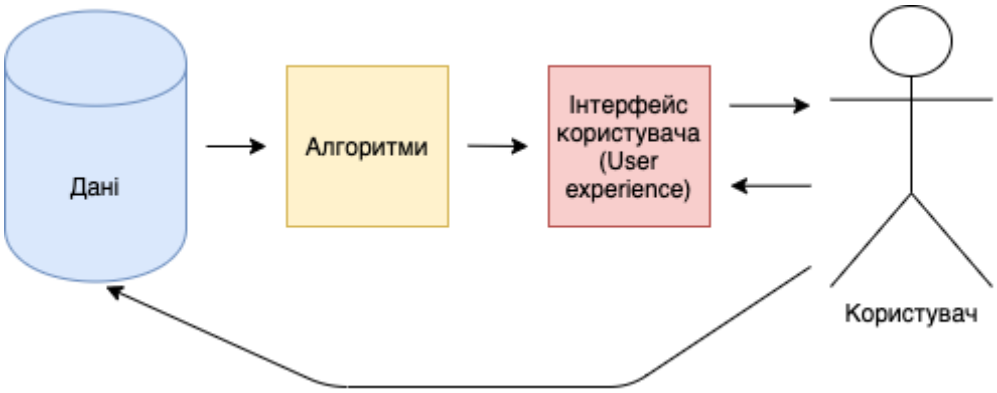


Рисунок. 1.1. Принцип роботи РС

Користувач залишає інформацію явно або неявно, система фільтрує і зберігає її. Збережені дані використовуються алгоритмами машинного навчання для тренуванн я моделей, використовуючи які можна передбачити успішні рекомендації для корист увачів.

1.2 Використання систем рекомендацій в електронній комерції.

Системи рекомендацій є одним із основних способів, за допомогою яких веб-сайти електронної комерції отримують постійних клієнтів. Повторна покупка - один з ключових показників для онлайн-магазинів, оскільки повторний клієнт означає менше грошей, витрачених на те, щоб спонукати його/її до покупки.

Системи рекомендацій - це не нова технологія. Вони з’явилися разом з інтернет-магазинами та стрімінговими сервісами. Сфери застосування рекомендаційних систем різноманітні: пошук фільмів, музики, наукових статей, роздрібна торгівля, соціальні мережі, електронна комерція, онлайн банкінг і т.д.

Мабуть, одним з найбільш широковідомих прикладів впровадження і використання рекомендаційних систем є компанія Netflix - постачальник відеоконтенту на умовах оренди і у вигляді потокового сервісу[5].

Компанія починала з того, що розсилала клієнтам по підписці VHS-касети і DVD. Користувач дивився і відправляв диски назад, отримував нові. Для Netflix було важливо підвищити якість рекомендацій. Чим краще Netflix рекомендує користувачам фільми, тим більше фільмів беруть в прокат. Відповідно, зростає і прибуток компанії.

У 2006 році компанія Netflix оголосила конкурс на вдосконалення своєї рекомендаційної системи під назвою Cinematch. Рекомендації формувалися з урахуванням як оцінок користувача, так і оцінок інших глядачів - для цього система підбирала користувачів зі схожими перевагами, чиї оцінки близькі до їх власним. На підставі цього глядачеві автоматично давалася рекомендація: подивитися той чи інший фільм. Власний алгоритм Netflix передбачав оцінок з якістю 0.9514 за метрикою RMSE. Завдання було поліпшити прогноз хоча б на 10% - до 0.8563. Переможцю обіцяли приз в $ 1 000 000.

Netflix виклав у відкритий доступ зібрані дані: близько 100 мільйонів оцінок за п'ятибальною шкалою з зазначенням ID користувачів, які поставили оцінку. Учасники змагання повинні були якомога точніше передбачати, яку оцінку поставить конкретному фільму той чи інший користувач.

Змагання тривало три роки. За перший рік якість поліпшили на 7%, далі все трохи сповільнилося. Проміжні номінації вручалися щороку до тих пір, поки дві команди з невеликою різницею в часі надіслали рішення, кожне з яких проходило поріг в 10%. Перший приз дістався компанії BellKor's Pragmatic Chaos, групі вчених з AT & T, яким вдалося домогтися поліпшення точності рекомендацій на 10,06%.

Компанії продають книги, підписки на музику, фільми, електроніку, побутові товари, автомобілі, нерухомість та багато інших товарів. Також, з’явилися такі платформи для онлайн навчання як Coursera, Udemy, Prometheus. Ці платформи пропонують велике різноманіття курсів на різні тематики, як платних так і безкоштовних. Очевидно, що кількість альтернатив продуктів, які може придбати покупець на веб-сайтах E-Commerce зростає. Сервіси Amazon, Alibaba пропонують сотні різних товарів, сучасні технології дозволяють обробляти набагато більшу їх кількість. При такому різноманітті користувачу складно підбирати підходящі товари, адже проаналізувати кожен з них фізично неможливо. Такі сервіси мають фільтри пошуку, що дозволяють шукати об’єкти за їхніми атрибутами, але не всі товари мають чітко-виділені атрибути. Наприклад, книги та музика. Добре спроектована рекомендаційна система вирішує цю проблему.

Рекомендацій в сфері електронної комерції представляють собою списки об’єктів виду top-N. Чим ближче до голови списку розміщений елемент, що рекомендується, тим більш релевантним він вважається для користувача[5].

На сьогоднішній день майже кожен інтернет-магазин використовує який-небудь тип рекомендацій, що і не дивно, оскільки ці системи можуть суттєво підвищити доходи, CTR, конверсії та інші важливі показники, якщо РС належним чином налаштована. Крім того, вони можуть мати значний позитивний ефект і на взаємодію з користувачем, що перекладається на показники, які важче виміряти, але, тим не менш, ці показники мають надзвичайно важливе значення для онлайн-бізнесу, наприклад, задоволення клієнтів та відсоток користувачів, які повертаються.

1.3 Вплив РС на аудиторію користувачів

Не потрібно проводити детальне дослідження ринку щоб побачити що клієнт буде більш готовий зробити покупку в магазині у якому він відчуває максимальну допомогу у пошуку потрібного продукту. Покупці також набагато частіше повертаються до такого магазину в майбутньому.

Кілька місяців тому Netflix підрахував, що його рекомендаційний двигун коштує 1 мільярд доларів США для компанії щорічно [5]. В даний час використання РС є абсолютно необхідною умовою успішного ведення бізнесу в Інтернеті. Найвідоміший приклад - статистичні дані Amazon, за якими 35% їх загальних доходів надходить від покупки продуктів, які клієнти знайшли за допомогою рекомендацій. Це дуже важливий показник при оцінці системи рекомендацій. Компанії не завжди враховують його. Нижче наведено кілька показників, які можуть описувати ефективність рекомендацій:

Коефіцієнт конверсії за рекомендаціями - показник переходів для користувачів, які натискали рекомендації.

CTR - відношення числа кліків на рекомендований продукт до числа його показів. Вимірюється у відсотках. Формула розрахунку: CTR = (число кліків / число показів) x 100 %. Наприклад, якщо рекламне оголошення було показано 100 раз, а

клікнули по ньому лише один раз, то показник CTR дорівнюватиме 1 %. Хоча це і очевидний, але дуже важливий показник, який слід враховувати.

% доходу отриманий від рекомендацій - це один з найбільш часто використовуваних показників (також висвітлених в прикладі Amazon вище). Це просто означає доходи через рекомендації / сукупні доходи.

Кількість переглянутих продуктів - кількість продуктів, які переглядають люди, які активно використовують рекомендації під час своїх сеансів. Хоча інтуїтивно більша кількість переглядів може також означати, що користувачу важко знайти потрібне, у дослідженні Wolfgang Digital прийшли до висновку, що час, витрачений на сайті користувачем, і кількість сторінок, які вони відкривають, корелюють з конверсією позитивно.[4]

1.4 Функції рекомендаційних систем

РС займають важливе місце в багатьох бізнес процесах. Саме тому багато великих компаній фінансово підтримують розвиток цієї сфери.

Серед основних функцій РС можна виділити наступні:

- *Збільшення кількості проданих товарів*. Це, ймовірно, найважливіша функція для комерційних РС, тобто, збільшити коефіцієнт продажу товарів за рахунок обігу рекомендованого товару у порівнянні з коефіцієнтом продажу без будь-якої рекомендації. Ці додаткові продажі досягаються тому, що рекомендовані елементи задовольняють потреби користувачів. Імовірно, користувач визнає це після того, як спробував кілька рекомендації. Загалом, можна сказати, що з точки зору провайдера, основною метою введення РС є збільшення кількості конверсій (conversion rate), тобто, кількості користувачів, які приймають до уваги цю рекомендацію і споживають рекомендований продукт.
- *Продаж більшої кількості специфічних предметів*. Ще однією важливою функцією РС є продаж предметів, які складно знайти без точної рекомендації через їхні унікальні характеристики, які до вподоби невеликій підмножині користувачів. Наприклад, у РС фільмів компанії Netflix, постачальник послуг

зацікавлений в оренді усіх DVD-дисків в каталозі, а не тільки найбільш популярних з них. Цього важко досягнути без РС, оскільки постачальник послуг не може рекламувати фільми, які, ймовірно, влаштовують смак невеликої групи користувачів. Таким чином, РС пропонує рекомендації непопулярних фільмів потрібним користувачам.

- *Підвищення задоволеності користувачів*. Добре розроблена РС може також поліпшити досвід користувача з сайтом або додатком. Поєднання ефективних, тобто точних рекомендацій збільшить суб'єктивну оцінку системи користувачем. Це, в свою чергу, збільшить використання системи та ймовірність того, що рекомендації будуть прийняті до уваги.
- *Збільшення довіри користувача*. Користувачі лояльні до веб-сайтів які пам'ятають попередню активність цього користувача. Це нормальна особливість РС оскільки багато алгоритмів у РС вирахують рекомендації, посилюючись на інформацію, отриману від користувача в попередніх взаємодіях, наприклад, його попередні рейтинги. Отже, чим довше користувач взаємодіє з сайтом, тим більш витонченою та персоналізованою для користувача стає модель [1].

1.5 Основні поняття в рекомендаційних системах

Користувач взаємодіє з системою через її *інтерфейс*, яких поділяється на *вхідний* та *вихідний*.

Вхідним інтерфейсом рекомендаційної системи називається сукупність точок взаємодії між користувачем і системою, через які система отримує персоналізовані дані користувача. Введемо поняття рейтингу. *Рейтинг* – якісна оцінка продукту користувачем, що, зазвичай, виражається числовим значенням. Наприклад, користувач може залишити оцінку відео, яке він переглянув, або залишити коментар під продуктом який він купив. Система отримує персональну інформацію від користувача двома способами: *явним* (explicit) і *неявним* (implicit). Вище наведені приклади *явного рейтингу* , адже користувач цілеспрямовано залишає оцінку, або коментар. Прикладом *неявного рейтингу* може послугувати ситуація, коли

користувач почав дивитися відео і закінчив перегляд за хвилину. Звідси система може зрозуміти, що відео користувачу не сподобалося, адже він завершив перегляд за хвилину.[6]

Вихідним інтерфейсом рекомендаційної системи називається сукупність точок взаємодії між користувачем і системою, через які користувач отримує рекомендації від системи.

Існує поняття *передбаченого рейтингу,* що визначає з певною ймовірністю передбачену оцінку деякого продукту користувачем.

Рекомендацією називають кінцевий набір предметів, який з певною ймовірністю буде до вподоби користувачу. Цей набір може бути відсортований у порядку спаду рейтингу.

1.6 Дані та джерела інформації для рекомендаційних систем

1.6.1 Типи даних

РС використовують дані різних типів, але їх можливо узагальнити до наступного вигляду: продукти та користувачі.

Продукти є предметами, які рекомендуються. Продукти можуть бути охарактеризованими їхніми властивостями та користю. Наприклад, одяг можливо поділити чоловічий та жіночий, він має колір та розмір. З іншої сторони користувач може бути задоволений, чи розчарований купленим одягом. Це також є характеристикою продукту. РС використовують ці характеристики по різному залежно від алгоритму. Найчастіше, дані про властивості продукта використовуються алгоритмами фільтрування за контентом (CBF), та алгоритмами кластеризації. Властивості продукта називають його ознаками, або опціями. Обов’язковим для продукта являється його ідентифікатор[6].

Користувач є об’єктом рекомендацій. Для того, щоб ці рекомендації були персоналізованими користувач має деякий профіль. Профіль користувача може містити різноманітну інформацію, як, наприклад, вік та стать. Профілем користувача вважається також список рейтингів, з якими він оцінив деякі продукти. Остання характеристика використовується алгоритмом колаборативного

фільтрування (CF). Обов’язковим для користувача є його ідентифікатор. На рисунку. 1.2 зображено відношення між користувачами та продуктами.

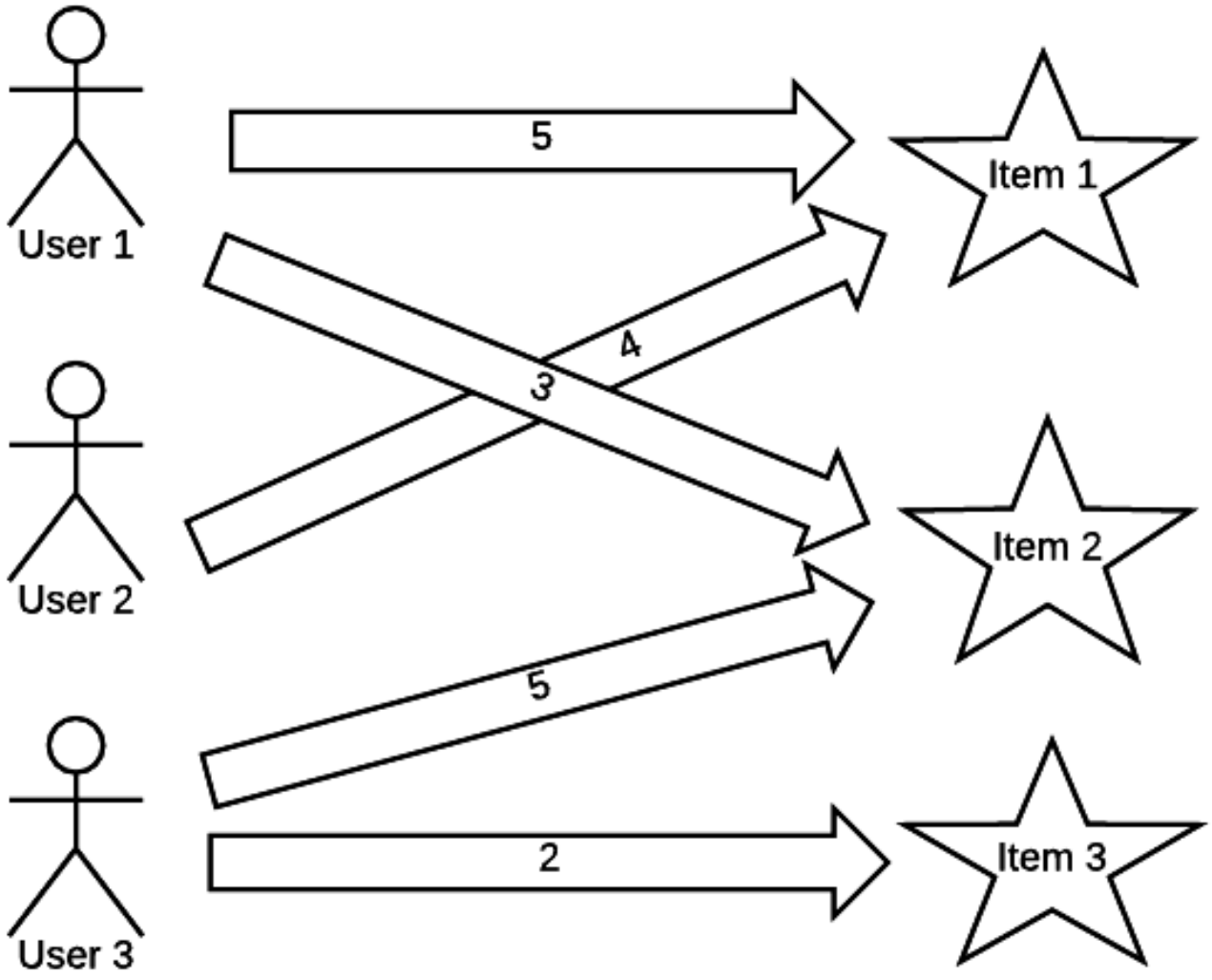


Рисунок. 1.2. Відношення між користувачами та продуктами

Користувачі купують продукти і залишають явний відгук про покупки за п'ятибальною шкалою. Користувач 1 оцінив продукт 1 та 2 з рейтингом 5 і 3 відповідно. Користувач 1 оцінив продукт 1 на 4. Користувач 3 оцінив продукт 2 та 3 на 5 та 2 відповідно. Ці дані рекомендаційна система використовує для підрахунку рекомендацій продуктів, які користувач ще не купував.

1.6.2 Джерела інформації в рекомендаційних системах

Дані, отримані для використання у рекомендаційних системах розділяють на *явні* та *неявні*. Явні дані збираються шляхом формування запитань для користувача, на які від дає відповіді. Це може бути його ініціатива, наприклад користувач залишає коментар, або оцінку продукту. Неявні дані збираються системою без потреби посередніх дій користувачем. Такі дані формуються поведінкою користувача на сервісі.

РС черпає явні дані для тренування моделей з набору рейтингів, характеристик продуктів та профілю користувачів. Рейтинги задаються у форматі, зображеному таблицею 1.1.

Таблиця 1.1 Список рейтингів до рисунку 1.2

Ідентифікатор користувача	Ідентифікатор продукта	Рейтинг
1	1	5.0
1	2	3.0
2	1	4.0
3	2	5.0
3	3	2.0

Зазвичай рейтинги використовуються алгоритмами колаборативного фільтрування та алгоритмами підрахунку неперсоналізованих рекомендацій. Продукт та його характеристики також несе важливу інформацію. Характеристики продукту, зазвичай, використовуються алгоритмами фільтрування за контентом (CBF) та алгоритмами кластеризації. Останні дають змогу підрахувати рекомендації групи продуктів, які має сенс купувати разом. Саме за допомогою цього алгоритму прославилася РС Amazon. Профіль користувача також може покращити точність рекомендацій. Наприклад, у випадку з одягом, інформація про стать користувача є надзвичайно важливою, адже це дасть змогу уникнути помилки рекомендації одягу для жінок чоловікам і навпаки.

Через те, що явних даних занадто мало по відношенню до кількості користувачів та продуктів сучасні підходи у більшості випадків використовують неявні дані. Вони задаються у вигляді журналу подій користувача на сервісі. Наприклад, користувач переглянув інформацію про продукт, користувач поклав продукт до кошика, користувач придбав продукт.

1.7 Підходи до реалізації рекомендаційних систем

1.7.1 Типи

Існує 4 типи рекомендаційних систем:

Коллаборативна фільтрація (Collaborative Filtering) - рекомендації засновані на історії оцінок як самого користувача, так і інших користувачів. Цей підхід має теоретично високу точність, але при цьому має одну важливу проблему - високий поріг входу.

Засновані на контенті (content-based) - рекомендації засновані на даних, зібраних про кожен конкретний товар. Користувачеві рекомендуються об'єкти, схожі на ті, якими він раніше цікавився або вже купував. Схожість оцінюється виходячи з вмісту об'єктів. Великий плюс - можливість зацікавити нового користувача пропозиціями з перших кроків. Для цього не потрібно довго збирати дані про переваги, а можна відразу включити споживача в роботу з ресурсом. Можливо рекомендувати навіть ті об'єкти, які не отримали оцінку інших користувачів. Основні недоліки - сильна залежність від предметної області, зниження точності і обмеженість корисності рекомендацій.

Засновані на знаннях (knowledge-based) - рекомендації засновані на знаннях про предметну область (а не про кожен товар). Такий тип рекомендацій має високу точність, пропонуючи користувачеві те, що йому потрібно. Крім цього, система вивчає і аналізує взаємозв'язки між об'єктами, враховує ряд додаткових опцій, що відносяться до індивідуальних властивостей конкретного користувача. До таких властивостей відносяться призначені для користувача побажання (наприклад, їх використовує Rozetka.ua) і демографічні особливості (вихідні дані, які

використовують найбільші соціальні мережі, такі як Facebook, LinkedIn, Вконтакте та інші). Основний мінус - складність розробки та збору даних.

Гібридні (hybrid) - рекомендації засновані на комбінуванні коллаборативного і тематичних підходів, що дозволяє уникнути більшості недоліків, властивих кожній системі.

Усі вище розглянуті підходи мають як переваги так і недоліки. Ці підходи детально порівнюються в наступному пункті [8].

1.7.2 Порівняння підходів до реалізації рекомендаційних систем

У цьому пункті порівнюються попередньо-розглянуті підходи до обчислення рекомендацій. Як було сказано, кожен підхід має свої переваги та недоліки. Нижче описуються деякі загальні проблеми, які будь-яка РС має вирішувати.

Розрідження матриці рейтингів. Матриця рейтингів – це матриця, де кожна колонка представляє продукт, а кожен рядок – користувача. Коли користувач **i** залишає рейтинг **r** для продукта **j**, то в матриці рейтингів елементу **ij** присвоюється значення **r**. Як правило, користувач залишає мало рейтингів для продуктів по відношенню до кількості продуктів. Саме тому матриця рейтингів є дуже розрідженою, що часто спричиняє проблему бракування інформації для знаходження користувачів з подібними уподобаннями. Ця проблема називається gray sheep problem.

Проблема холодного старту. Ця проблема виникає у CF алгоритмах, коли додається нова сутність в систему. Новий продукт не може бути рекомендованим з самого початку, коли він додається в систему і ще не має жодного рейтингу. MovieLens (movielens.org) не може рекомендувати нові відео, поки вони не отримають деякі початкові рейтинги. Проблема нового користувача є дещо складнішою для вирішення, адже неможливо знайти схожих користувачів, або створити профіль для CBF алгоритма, допоки цей користувач не залишив хоча б одного рейтингу.

Проблема масштабованості. Ця проблема виникає завжди, коли об’єм даних зростає. У реальних РС Netflix та Amazon алгоритми є масштабованими. У найкращому випадку обчислювальна складність алгоритму має лінійно зростати зі збільшенням кількості даних. Також деякі алгоритми дають високу точність на невеликих наборах даних, але на реальних даних якість рекомендацій значно падає.

Проблема конфіденційності. Щоб обчислювати якісні рекомендації РС має потребу збирати якомога більше даних про користувачів. З іншої сторони користувач може мати підстави підозрювати, що система знає занадто багато про нього. Цей аспект негативно впливає на подальше користування сервісом. Саме тому РС повинна ненав’язливо і прозоро збирати інформацію про користувача, даючи можливість надати більше даних, якщо цього бажає користувач.

Міцність системи. Під цим терміном розуміють можливість системи розпізнати фальшиві транзакції користувачів, спрямовані на те, щоб зробити деякі продукти більш популярними ніж інші.

Це спричиняє велике обчислювальне навантаження.

На практиці комбінують підходи до обчислення рекомендацій, адже це дає змогу пок рити недоліки одного алгоритму перевагами іншого.

Рекомендаційна система є необхідним інструментом для фільтрування непотрібних або нецікавих сутностей тому чи іншому користувачу. Наразі РС використовуються багатьма комерційними компаніями. Вони спрощують життя користувачам інтернет-магазинів, та є основним джерелом збільшення кількості продаж продуктів.

2. АЛЬТЕРНАТИВНІ ПІДХОДИ ДО РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Гібридні рекомендаційні системи

Гібридні рекомендаційні системи об’єднують два, або більше рекомендаційних підходів, щоб отримати більшу точність та продуктивність, покриваючи недоліки кожного окремого підходу в системі. Найчастіше підхід колаборативної фільтрації (CF) поєднується з деяким іншим підходом, щоб уникнути проблему надмірної специфічності рекомендацій[8]. Таблиця 2.1 показує методи комбінування різних підходів до обчислення рекомендацій.

Таблиця 2.1 Порівняння рекомендаційних підходів

Гібридний підхід	Опис
Зважений (Weighted)	Рейтинги, що передбачені різними алгоритмами лінійно, або нелінійно комбінуються в єдине значення, яке й використовується для вибору продуктів для рекомендації.
Перемикання (Switching)	Система вибирає підхід, що використовується для підрахування рекомендації залежно від ситуації.
Мікс (Mixed)	Рекомендації з декількох різних підходів змішуються в один список
Комбінування властивостей (Feature combination)	Властивості від різних джерел даних комбінуються і передаються як вхідні дані до деякого алгоритму
Каскадування (Cascade)	Один підхід уточнює результати рекомендацій іншого

Зважений підхід (Weighted)

У зваженому гібридному підході оцінка рекомендованого продукту обчислюється виходячи з результатів усіх доступних методів рекомендацій, що присутні в системі. Перевага зваженої гібридної системи в тому, що всі її можливості задіяні у просту та зрозумілу лінійну комбінацію результатів. Зміну коефіцієнтів можна виконувати за лічені секунди.

Перемикання (Switching)

Перемикальна система використовує деякий критерій для визначенні проміжків часу або ж певних моментів, коли той чи інший підхід буде давати кращі результати.

В рамках перемикальної системи ми знову незалежно будуємо наші рекомендаційні системи, і далі, в той момент, коли нам потрібно обчислити рекомендації для користувача, ми вибираємо, яка рекомендаційна система буде це робити. Тобто в рамках цього підходу ми вже не змішуємо разом рекомендації від різних систем, а вибираємо, яка система буде джерелом рекомендацій для користувача. Для того щоб зробити такий вибір, нам потрібно побудувати деякі критерії, які підкажуть нам, яку систему слід вибрати. Ідея тут досить проста. Наприклад, CBF підхід дозволяє нам вирішувати проблему холодного старту для нових продуктів. Тому якщо ми хочемо порекомендувати користувачеві об'єкт, якого в нашій системі ще не було або він з'явився недавно, і досить багато статистики про нього ми ще не накопичили, то в цьому випадку нам логічніше використовувати CBF підхід. Якщо ж ми намагаємося оцінити продукт, який вже давно в нашій системі та є багато рейтингів цього продукту, то CF, можливо, в цьому завданні спрацює краще. Відповідно, ми можемо побудувати деякі критерії, на підставі яких ми будемо робити вибір, і використовувати правильну рекомендаційну систему.

Мікс (Mixed)

В рамках цього підходу ми знову генеруємо рекомендації незалежно, кожна система будує свій список рекомендацій, однак підсумковий список будується на основі суміші рекомендацій від різних систем. Це добре працює в тих випадках, коли нам потрібно одночасно отримати довгий, різноманітний список рекомендацій, наприклад, безперервний feed новин або рекомендації медіаконтенту. В даному випадку часто використовують цю схему, тому що ми можемо доповнювати рекомендації від різних рекомендаційних систем один одним.

Комбінування властивостей (Feature combination)

Цей підхід в деякій мірі заснований на підході CBF і є його доповненням. Ідея в тому, що ми хочемо об'єднати в одній навчальній вибірці ознаки від різних підходів.

Найчастіше це виглядає наступним чином: ми використовуємо інформацію про переваги схожих користувачів, отриману в рамках коллаборативного підходу (CF), в якості ознак для CBF підходу. Тобто ми з вами можемо розглянути рейтинги, які отримав об'єкт за допомогою користувачів, схожих на нашого користувача, і цю інформацію використовувати як ознаки. Таким чином, ми збагачуємо ознаки опису об'єкта, отриманого в рамках content-based підходу, і додаємо туди інформацію про рейтинги. Це часто досить непогано працює[8].

Каскадування (Cascade)

Це дуже красива ідея, в рамках якої ми послідовно застосовуємо декілька рекомендаційних систем для уточнення рекомендацій. Припустимо, що у нас є рекомендаційна система, яка працює швидко і може досить грубо і просто поділити об'єкти на дві множини: точно нерелевантні, тобто нецікаві користувачеві і потенційно релевантні. Тоді давайте застосуємо цю рекомендаційну систему, відкинемо ті об'єкти, які нашого користувача точно не зацікавлять, і далі будемо працювати тільки з тією групою об'єктів, які потенційно релевантні. Ось до них давайте застосуємо більш складну рекомендаційну систему, яка, можливо, працює довше, але зате вона зможе більш точно відранжувати ці об'єкти для користувача. Така схема часто називається Candidate selection, коли за допомогою одного алгоритму, що виконує обчислення більш ефективного, ми з відбираємо безліч кандидатів і далі за допомогою іншої, більш складної, системи їх ранжуємо.

2.2 Рекомендації на типовому веб-сайті електронної комерції

Є багато місць, де рекомендації можуть відображатися на сайті електронній комерції. Основні методи рекомендацій найкраще працюють на головних сторінках, у кошиці, на сторінках категорії.[4]

2.2.1 Рекомендації на головній сторінці

Головна сторінка - це перше, що бачить користувач коли відвідує веб-сайт через прямий трафік. Оскільки ці відвідувачі не обов'язково шукають щось

конкретне, рекомендації на головній сторінці мають на меті інформування клієнтів про останні пропозиції та знижки та демонстрацію своїх популярних продуктів.

Популярні продукти. Це дуже базова, але потужна логіка рекомендацій, яка чудово працює майже у всіх магазинах електронної комерції. Популярність продукту визначається найлегше за кількістю придбаних вами товарів. Проте, більш досконалі системи включають інші дані (кліки, перегляди, тощо) у свою логіку, щоб подати ще більш точні рекомендації.

У випадку із змістовними сайтами (новинними сайтами, відеопорталами) інші чинники, такі як час, витрачений на сторінку, прокручуваний відсоток, або час перегляду відео, також можуть бути важливими факторами популярності. На веб сайтах, що пропонують певну продукцію ця логіка простіша. На рисунку 2.1 приведено приклад рекомендації бестселлерів [26].

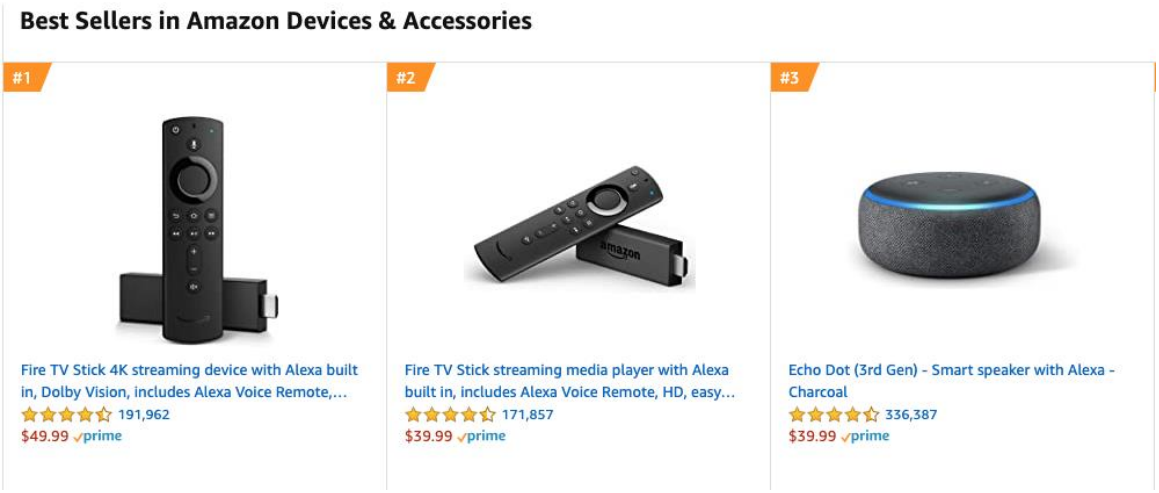


Рисунок. 2.1. Приклад рекомендації бестселерів

Рекомендації, що базуються на рейтингу. Іншим показником популярності може бути високий рейтинг продукту. Під рейтингом мається на увазі “явний рейтинг”: оцінки, коментарі. З іншого боку, перегляд опису товару, покупки та перегляди називаються "неявним зворотним зв'язком", оскільки вони представляють точки даних, які створюються, коли користувач природно взаємодіє з платформою чи веб-сайтом. Системи рекомендацій, розгорнуті в реальному світі (а не в академічному), використовують неявні відгуки через велику кількість, багатство та структуру таких даних, на відміну від явних рейтингів, наприклад, які є надзвичайно

дефіцитними (лише невеликий відсоток користувачів пише відгуки або навіть дають рейтинги). Тим не менш, хоча і веб-сайт чи платформа, можливо, не мають великої користі від рейтингів, користувачам це допомагає. Насправді, згідно з опитуванням BrightLocal, 88% користувачів довіряють рейтингу так само, як і особистим уподобанням.

Інколи, з огляду на звички та уподобання бази користувачів, відгуки можуть стати вирішальним чинником впливу на рішення про покупку. Тоді, позиції "Найвищий рейтинг" розміщуються також на головній сторінці. На рисунку 2.2 приведено приклад рекомендації товарів з найвищим рейтингом [27].

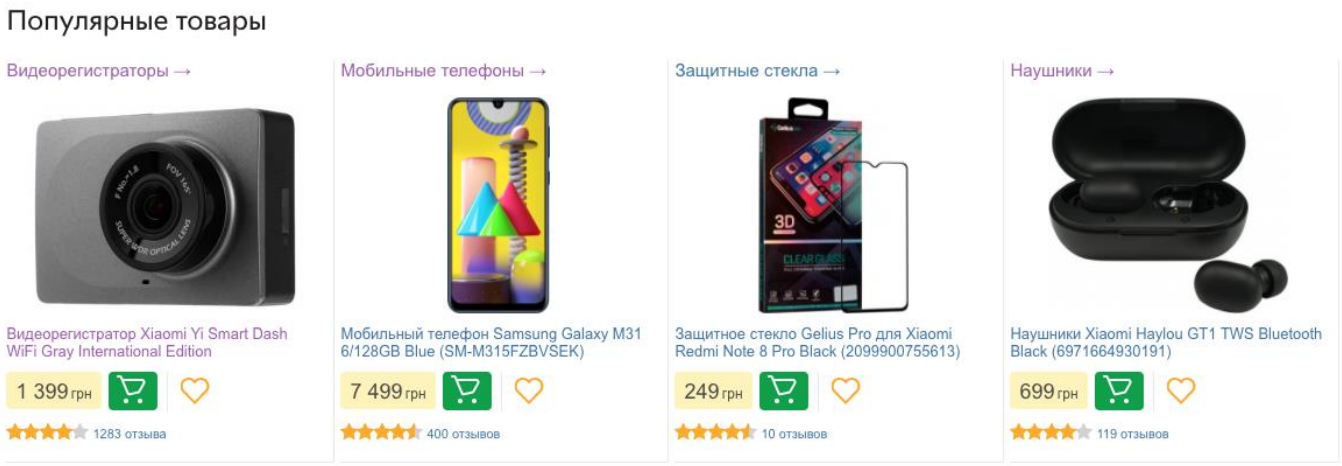


Рисунок. 2.2. Приклад рекомендації товарів з найвищим рейтингом

Персоніфіковані рекомендації. Списки персоналізованих рекомендацій показують різні продукти для кожного користувача, залежно від їх попередньої покупки та історії веб-перегляду. Проте персоналізовані рекомендації потребують значних обсягів поведінкових даних користувачів, яких система не має для нових відвідувачів (це називається проблемою холодного старту РС).

Для того, щоб забезпечити персоналізований користувацький досвід для користувачів які відвідують веб-сайт не вперше, а також задовольняти нових відвідувачів, найкращим рішенням у галузі є визначення "альтернативного сценарію" у випадку, коли про користувача мало що відомо, або невідомо нічого. На практиці це означає, що система виявляє, чи є достатньо даних про певного клієнта для

надання йому персоналізованих пропозицій предметів. Якщо даних немає, система рекомендацій "повернеться" до більш загальних рекомендацій (наприклад, популярні продукти) або логіки, яка фільтрує продукти за категорією, відображаючи в рекомендаціях тільки продукти з категорії, в якій були товари переглянуті користувачем в даній сесії.

Сценарій альтернативних рекомендацій слід планувати з великою обережністю і використовувати максимально ефективні дані для того, щоб забезпечити точні рекомендації для нових відвідувачів. На рисунку 2.3 приведено приклад персоніфікованих рекомендацій [26].

Рисунок. 2.3. Приклад персоніфікованих рекомендацій

2.2.2 Рекомендації на сторінці продукту

Сторінка продукту чи сторінка інформації про продукт - це місце, де відвідувачі можуть знайти детальний опис продукту та його характеристик. На цій сторінці також зазвичай є можливість додати продукт до кошика або замовити його відразу. Основна мета рекомендацій на цих сторінках - показати найбільш релевантні елементи, щоб, таким чином, забезпечити "наступний крок" у пошуку користувачем товарів та заохотити його на наступні покупки на веб-сайті.

У більшості випадків, чим більше часу користувачі витрачають у інтернет-магазині, тим вище шанси на те, що вони будуть купувати товари.

Схожі продукти. Віджети зі списком схожих продуктів можуть базуватися на дуже різних даних. Найменш складним є проста фільтрація на основі категорій, яка може бути застосована навіть без рекомендаційного механізму (це, безперечно, відстає в продуктивності). Якщо поєднати цей простий метод фільтрування із

подібністю на основі метаданих (описи, заголовки продуктів, теги, ціни тощо), можна значно підвищити продуктивність (наприклад, рекомендувати елементи однієї марки або того самого кольору з поточного категорії), але для цього потрібно буде мати розширені функціональні рекомендації, доступні на сайті. Одним з найефективніших реалізацій на основі подібності є метод, що називається "спільна фільтрація між елементами" (метод, який був започаткований компанією Amazon). На рисунку 2.4 приведено приклад рекомендації схожих продуктів [26].

Рисунок. 2.4. Приклад рекомендації схожих продуктів на сторінці продукту

"Клієнт, який купив / переглянув це також купив" Колаборативна фільтрація. Колаборативна фільтрація в рекомендаціях щодо продуктів електронної комерції була вперше застосована Amazon (вони створили свій початковий патент для колаборативної фільтрації між елементами ще в 1999 році). Ідея коллаборативної фільтрації проста - якщо користувач здійснював покупки товару або перегляди контенту, знайдемо користувачів зі схожими смаками, і порекомендувати нашому клієнтові те, що схожі на нього люди споживали, а клієнт ще ні. Це User-Based підхід.

Аналогічним чином можна подивитися на завдання з точки зору товару, і підібрати компліментарні товари до кошика клієнта, підвищивши середній чек, або замінивши відсутні на складі товари на аналог. Це Item-Based підхід[3]. На рисунку 2.5 приведено приклад колаборативної фільтрації [26].

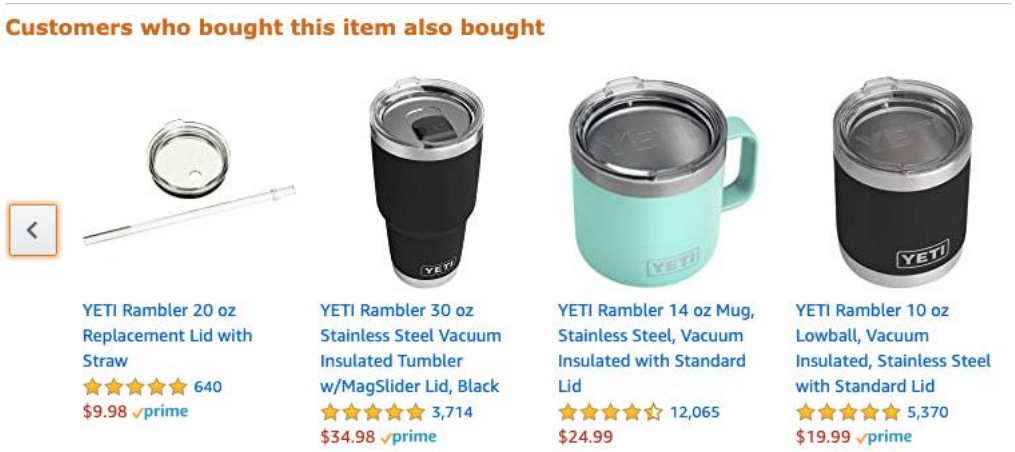


Рисунок. 2.5. Приклад рекомендації колаборативної фільтрації

Item-to-item collaborative filtering рекомендації: Цей тип логіки рекомендацій в основному визначає подібність двох продуктів, переглядаючи, як часто вони присутні разом у історії перегляду чи покупки користувачів. На практиці віджети з використанням такого роду рекомендацій мають назву, яка пояснює логіку рекомендацій: "Клієнти, які переглядали це, також переглянули ...". Моделі уподобань користувачів, які будуються цим алгоритмом, є дуже достовірними і специфічними до веб-сайту ваших користувачів, оскільки вони визначають кількісно реальні взаємодії між відвідувачами та платформою [8].

Персоналізовані рекомендації. Враховуючи контекст (продукт, який переглядається), разом з історією поточного користувача, рекомендації на сторінках продукту можуть бути персоналізовані ще більше. Для цього сценарію можна застосувати персоналізовані алгоритми колаборативного фільтрування. На рисунку 2.6 приведено приклад персоналізованої рекомендації [26].

Рисунок. 2.6. Приклад персоналізованої рекомендації

Personalized collaborative filtering рекомендації: Найпоширеніший спосіб персоналізованих рекомендацій - фокусуватися на середній подібності продуктів до останньої кількості продуктів, які користувач переглядає. Система використовує такий самий алгоритм знаходження подібності продуктів, але вона не порівнює кожен елемент з якимось одним, а з усіма елементами історії користувача. Зважування вихідного потоку залежно від часу взаємодії зазвичай підвищує точність.

2.2.3 Рекомендації для кошика

Рекомендація подібних або пов'язаних продукти з вмістом кошика відвідувача на сторінці кошика може бути дуже ефективним способом підвищення кількості продажів.

Рекомендації для кошика досягають клієнта в сприятливому психологічному стані, при якому він, швидше за все, вже вирішив здійснити покупку, тому він буде більш готовий відповісти «так» на подальші пропозиції. Існує декілька логічних рекомендацій, які дуже підходять для цього завдання. За дослідженням 2015 року, проведеного в 300 магазинах електронної комерції компанії Braillance та Marketingsherpa, віджети, які рекомендують додаткові продукти до елементів у кошику користувача, були одними з 10 найбільш ефективних типів рекомендацій, базуючись на прибутку від таких рекомендацій.

Рекомендовані аксесуари. Рекомендовані аксесуари для продуктів можуть суттєво збільшити розмір та вартість замовлення на веб-сайті. Більше того, реалізація таких віджетів може бути досить простою, технічно кажучи. Однак, залежно від розміру каталогу та структури категорії, воно може бути більш довгим при втіленні, оскільки дуже важко автоматизувати процес рекомендації сумісних аксесуарів для кожного товару. Тому такі логіки, як правило, задаються вручну пов'язаними між собою категоріями.

Оптимально, рекомендувати одну категорію іншій безпосередньо (наприклад, iPhone -> Аксесуари для iPhone). Це набагато простіше для автоматизації та масштабування, але вимагає надзвичайно продуманого планування структури категорій. Одним з таких легко кодифікованих правил, які можуть визначати

аксесуари по відношенню до продукту, є предмети, які часто купуються разом із ним, але коштують значно (в 2, 3, 10 разів) менше. Цей метод часто використовується, коли даних забагато для описування правил вручну. На рисунку 2.7 приведено приклад рекомендації товарів пов'язаних категорій [27].

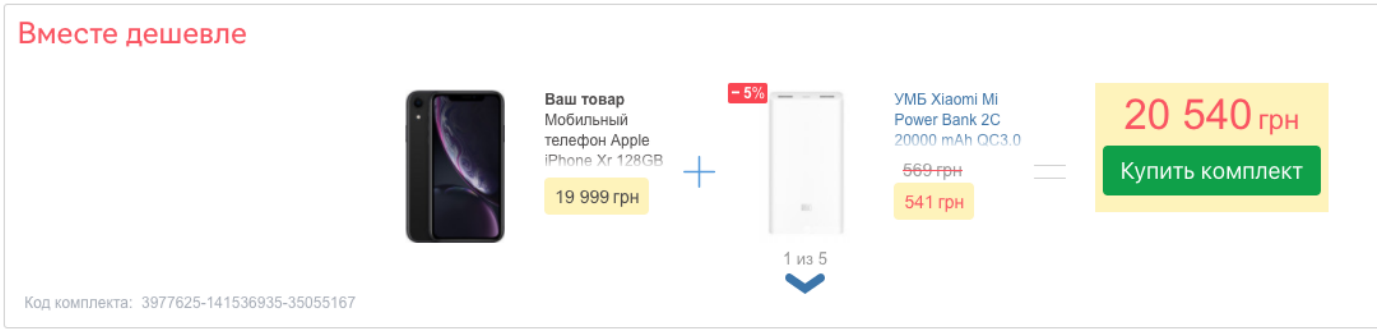


Рисунок. 2.7. Приклад рекомендації товарів пов'язаних категорій

Часто купується разом. Відображення часто продаваних продуктів на сторінках кошика може бути дуже ефективним. Макет цієї сторінки також є важливим фактором. Якщо у вас є час і ресурси, тестування різних планів та дизайнів A / B може дати чудові результати та статистичні дані. Наприклад, Netflix широко використовує тести A / B для макетів домашнього екрана на різних платформах. На рисунку 2.8 приведено приклад рекомендації товарів які купуються разом з даним товаром [27].

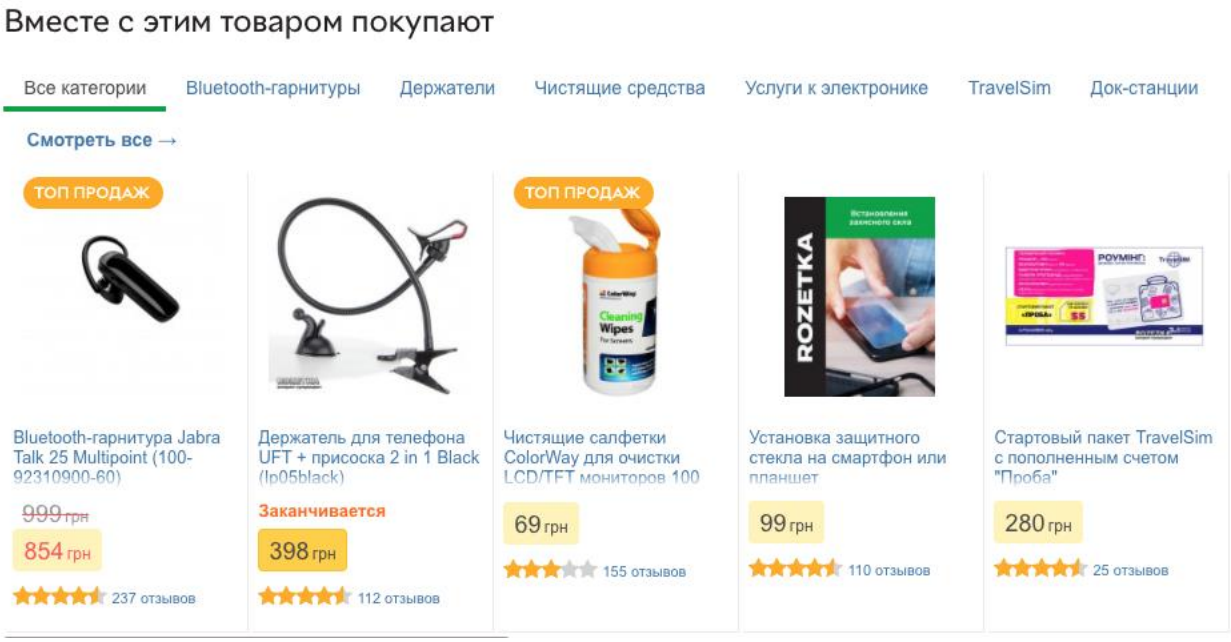


Рисунок. 2.8. Приклад рекомендації товарів які купуються разом з даним товаром

2.2.4 Рекомендації на сторінці категорій

Коли користувач переглядає сторінку категорії, він вже дав вам цінну інформацію про те, що він шукає. Тут ваша мета повинна полягати в наданні допомоги, тому йому не доведеться переглядати всю категорію, перш ніж надходити до потрібного товару або вмісту. Незважаючи на те, що найкращим варіантом у цьому сценарії є передовий фільтр та графічний пошук, рекомендації також можуть забезпечити цінність у цих випадках.

Нульовий результат і сторінка 404. Перехід до "нульового результату пошуку" або сторінка 404 часто означає завершення сеансу клієнта на сайті та, таким чином, потенційну втрату конверсії. Ці сторінки найчастіше є тими, що мають найвищий рівень виходу, а також однією з небагатьох, де швидкість виходу дійсно є дуже важливим показником.

Поле пошуку на 404 сторінках іноді може компенсувати UX. Відображення рекомендацій персоналізованих продуктів на цих сторінках також може допомогти клієнту повернутися до попередньої сторінки. На рисунку 2.9 приведено приклад списку переглянутих товарів на сторінці 404 [27].

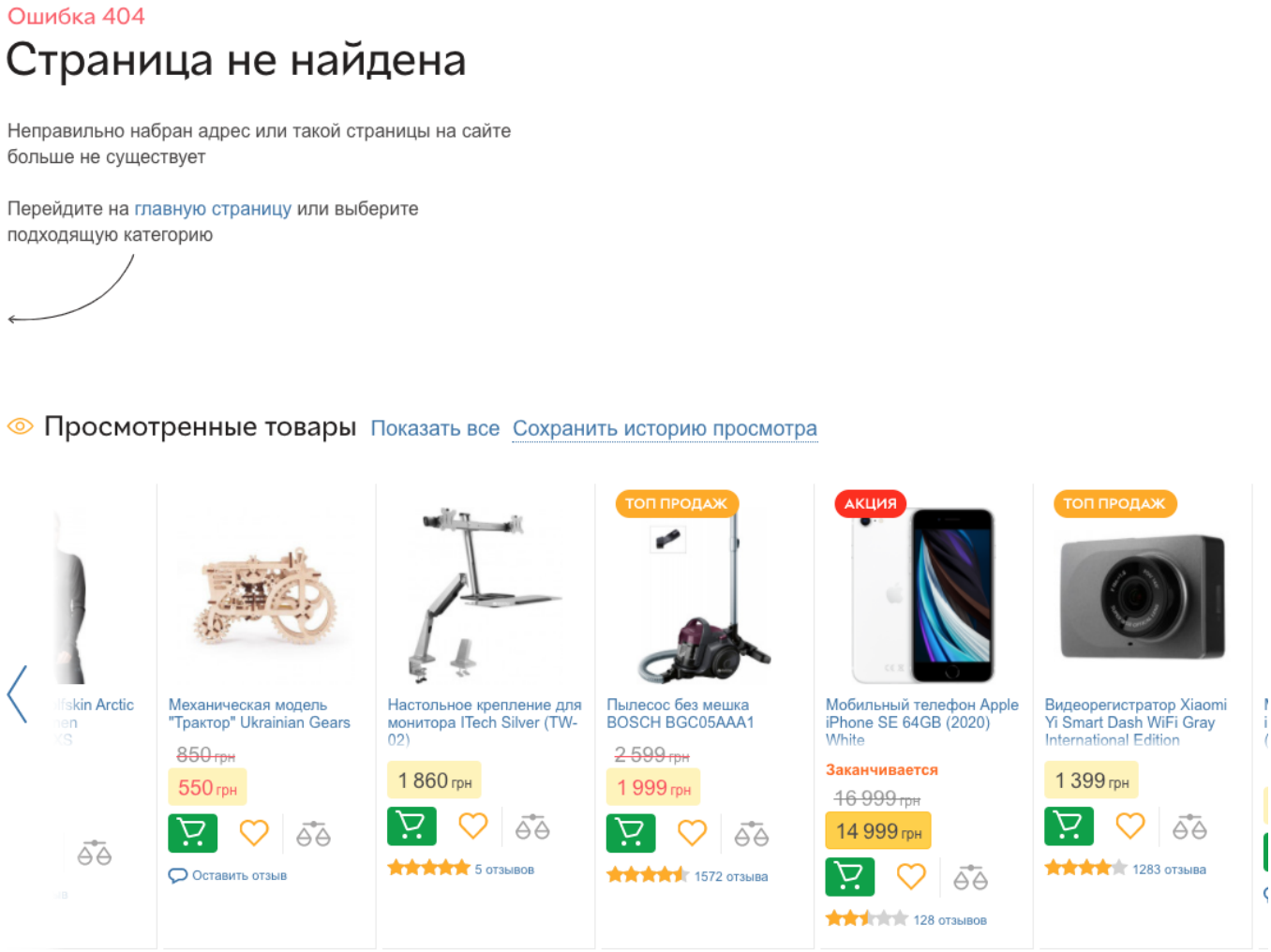


Рисунок. 2.9. Приклад списку переглянутих товарів на сторінці 404

2.3 Метрики рекомендаційних систем

Будь-яка рекомендаційна система допомагає вирішувати певну бізнес-задачу. А результат вимірюється особливими для конкретного бізнес-завдання способами: збільшенням кількості відвідувачів, кількості продаж і т.д. Однак якість рекомендаційного алгоритму таким способом виміряти занадто складно, адже бізнес-метрики залежать від величезної кількості умов, серед яких рекомендаційний алгоритм сам по собі може виявитися десятою справою. Точність рекомендаційних алгоритмів досить складно оцінити, але існують деякі стандартні метрики, що дозволяють описати точність з математичної точки зору.

Оптимальний підхід, про який піде мова - це спробувати створити метрику безпосередньо під бізнес-завдання. Далі мова піде про конкретний приклад: як це

зроблено для рекомендаційного сервісу кінофільмів imhonet.ru. Хоча цей підхід був перевірений саме на фільмах, але він досить загальний, в чому можна переконатися, замінивши слово фільм на що-небудь ще.

В задачі мова йде саме про список рекомендацій, а не про один елемент. Це тому, що система не в змозі отримати всю інформацію про бажані результати. Навіть пошуковик видає список посилань: він не знає повного контексту запиту. Якби він знав повний контекст (наприклад, повну інформацію в голові користувача), то, напевно, єдиний результат мав би сенс, а інакше частина роботи все одно залишається на плечах автора запиту. На рисунку. 2.10 зображено можливі варіанти реакції користувача на рекомендації.

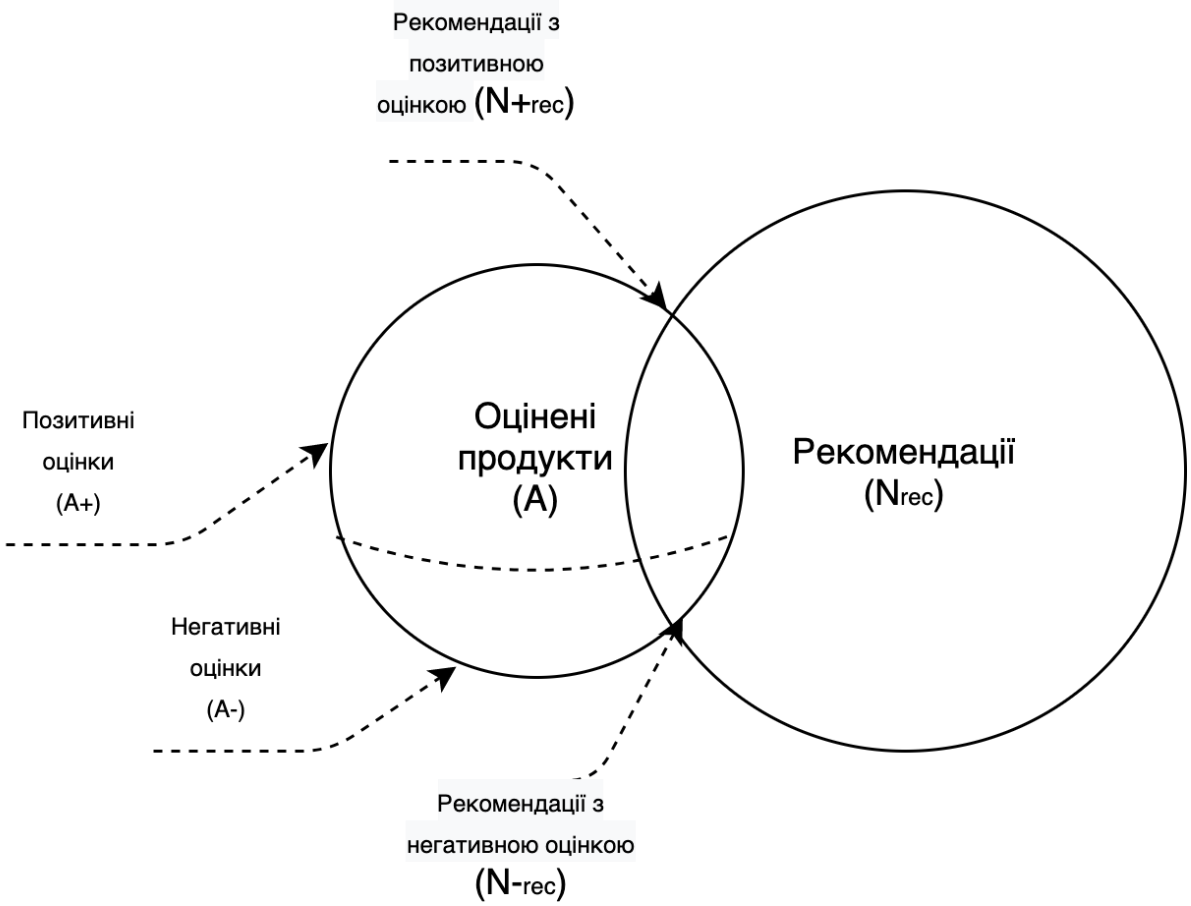


Рисунок. 2.10. Реакція користувача на рекомендації

Розглянемо 4 наступні ситуації. Hit (A+) – фільм рекомендувався і користувач відреагував на нього позитивно. Mishit (A-) – фільм рекомендуваався, але реакція користувача на нього була негативною. Recommended but not rated – фільм

рекомендувався, але оцінка користувача невідома (Nrec \ A). Rated but not recommended – Користувач позитивно відреагував на фільм, але цей фільм не рекомендувався (A+ \ Nrec).

Hit. Якщо певний фільм рекомендувався користувачу, після чого користувач залишив позитивну оцінку про фільм, значить, ми поцілили точно. Саме такі випадки нас і цікавлять, тому назвемо їх успіхами і будемо максимізувати їхню кількість.

Mishit. Фільм, який рекомендувався і отримав негативну оцінку є прикладом випадків, яких ми повинні уникати. Можливо навіть, що уникнення таких випадків важливіше, ніж максимізація кількості випадків успіху. Але, практика показує, що такими випадками можна просто знехтувати, оскільки ймовірність збігу негативних оцінок елементів зі списку рекомендацій, яку дає певна РС, виявляється надто незначною, щоб помітно вплинути на значення метрики. Тому, до негативних оцінок метрика буде нечутлива. Точніше, негативна оцінка буде еквівалентна просто відсутності оцінки у користувача.

Recommended but not rated. Якщо фільм рекомендувався, але позитивного відгуку у нас немає, то здається, що це нічого не означає. Людина могла не подивитися фільм або не оцінити його. На практиці виявляється, що таких випадків більшість. Тому, по-перше, ми завжди повинні рекомендувати з деяким надлишком (тобто взяти значення N_{rec} з запасом), по-друге, більшість людей, як правило, не оцінює кожен переглянутий фільм[14].

Rated but not recommended. Протилежний випадок – користувач позитивно оцінив фільм, який не рекомендувався. Тобто РС упустила можливість збільшити кількість успіхів. Поки такі можливості зустрічаються, можна покращувати точність РС.

Якщо всі елементи рекомендації містять позитивні оцінки, точність системи максимальна - покращувати більше нічого. На рисунку 2.11 зображено діаграму рекомендацій для ідеальної РС.

Рисунок. 2.11 РС з ідеальною точністю

Отже, поліпшення точності рекомендацій - це збільшення кількості успіхів в списку рекомендацій N^{+}_{rec} (розмір якого, N_{rec} фіксований).

Метрика precision

Якщо замість кількості успіхів ми будемо використовувати значення precision (p), тобто розділимо N^{+}_{rec} кількість успіхів на N_{rec}, то, по суті, нічого не зміниться: замість кількості успіхів будемо максимізувати величину, яка відрізняється від нього лише розподілом на константу:

$$p = \frac{N +_{rec}}{N_{rec}}$$

Зате, як ми побачимо трохи пізніше, цей поділ дозволить врахувати в метриці дуже суттєвий аспект завдання - залежність метрики від порядку рекомендацій в списку. Крім того, у значення p є зрозуміла логіка, яку можна описати імовірнісною мовою. Припустимо, що людина споживає наш продукт, а саме, переглядає всі елементи списку рекомендацій. Тоді p означає для нього ймовірність знайти в цьому списку потрібний елемент - такий, який задовольнить його в відносному майбутньому. Позначимо precision для користувача u як p_u:

$$p_{u=} \frac{N+_{rec}(u)}{N_{rec}}$$

Доречно узагальнити цю формулу на всю нашу аудиторію (або вимірювану вибірку) користувачів (Users):

$$P_{Nrec} = \underset{u \in Users}{mean(p_u)} = \frac{1}{|Users|} * \sum_{u \in Users} \frac{N^+_{rec}(u)}{N_{rec}} = \frac{N^+_{rec}}{N_{rec} * |Users|}$$

Кожен дивиться свій власний список і вибирає те, що йому потрібно, а P_{Nrec} показує середню ймовірність успіху для всіх випадків. Значення в правій частині - це тепер загальна кількість успіхів у всій вибірці.[14]

Дисконтування та метрика AUC

До сих пір ми оцінювали список елементів як ціле, але ми прекрасно знаємо, що його початок важливіший, ніж хвіст. У всякому разі, якщо список не надто короткий. Метрика, яка це враховує, а значить залежить від порядку елементів у списку, називається дисконтованою. Початок списку важливіше, ніж хвіст, через нерівномірний розподіл уваги користувачів - люди майже завжди дивляться на перший елемент, дещо рідше дивляться і на перший і на другий і т.д. Це означає, що для правильного дисконтування нам потрібна деяка поведінкова модель. Ми уявляємо собі усередненого користувача, який переглядає елементи по черзі, починаючи з першого, а потім, в певний момент, перестає це робити.

Ми не можемо ідентифікувати конкретного користувача, та й не хочемо. Іноді один і той же чоловік захоче переглянути список з 2 елементів, а іноді з 20. Було б непогано роздобути середні ймовірності переходу від кожного елемента до наступного, але нам вистачить даних навіть трохи простіших.

Припустимо, що у нас є ймовірність того, що довільна людина перегляне список з N елементів: w_N для будь-якого розумного N. Тоді можна за формулою повної ймовірності усереднити значення P_N, розуміючи під P_N середнє значення ймовірності успіху саме для тієї частини нашої аудиторії, яка переглянула список довжиною N. У нашій термінології це буде виглядати так:

$$AUC = \sum_{N_{rec}=1..\infty} w_{N_{rec}} \cdot P_{N_{rec}}$$

Виходить, що значення AUC оцінює середню ймовірність успіху персональних списків рекомендацій в більш реальних умовах - коли списки переглядають живі люди, увагу яких розподілено нерівномірно. Зауважимо, що щоб отримати величину AUC, ми використовували залежність величини P_{Nrec}(precision) від N_{rec} - розміру списку рекомендацій, який трохи раніше ми фіксували [14].

2.4 Рекомендаційна система на основі алгоритму CCO (Correlated Cross-Occurrence)

У сфері електронної комерції часто явні відгуки користувачів про продукти, або їх враження у іншому вигляді, що були явно сказані є недоступними. Навіть, якщо дані явного відгуку є доступні, їх кількість надзвичайно мала по відношенню до кількості користувачів та кількості продуктів. Кажуть, що матриця Користувач-Продукт є розрідженою.

Сфера електронної комерції характерна великими об’ємами даних поведінки користувача. Саме поведінка може найчастіше сказати що користувачу подобається, аніж сам користувач. З даних поведінки користувача в сервісі електронної комерції можна дістати дані неявного відгуку, їх ще називають індикаторами. Індикатор вказує на певний рівень вподобання продукту користувачем з певною ймовірністю. З поведінкових даних користувача можна, як правило, розпізнати дані більш як одного типу індикатора.
Рекомендаційна система, яка правильно використовує дані усіх доступних типів індикаторів вважається якісною.

Розглянуті у цьому розділі гібридні рекомендаційні системи працюють переважно з даними явного відгука, або ж потребують переведення даних явного відгука в дані неявного відгука, що не є тривіальною задачею, коли типів таких відгуків більше ніж 1. У останньому випадку припускають, що кожен тип індикатора має свою вагу, наприклад: купівля – 5, додавання товару до кошика – 3, перегляд – 1. Але це не завжди вірно. Серед таких даних багато шуму, особливо у випадку індикаторів з

низьким пріоритетом, наприклад, як перегляд товару. Дані потрібно фільтрувати. Алгоритм Correlated Cross-Occurrence призначений для створення моделей РС для даних багатьох типів індикаторів. Персоналізовані рокемендації можна обчислити за формулою:

$$r = (\mathrm{P}^T P)h_p$$

r – рекомендації,

h_p- дані історії користувача певного типу індикатора (купівля продукту, наприклад),

P - матриця історія усіх користувачів головного типу індикатора, рядки є користувачі, кононки є продуктами.

У даній формулі вираз ($\mathrm{P}^T P$) порівнює продукти між собою використовуючи (Log Likelihood Ratio) LLR кореляційний тест. Назвемо матрицю ($\mathrm{P}^T P$) матрицею індикатора головної події (наприклад, купівля). Рядками та колонками у цій матриці є продукти, а елементами similation/correlation оцінка. LLR знаходить аномальні патерни купівлі продуктів разом, фільтруючи події з історії користувача, які не надають користі в обчисленні рекомендацій.

Вираз вище можна інтерпретувати наступним чином. Береться історія користувача h_p та порівввнюється з рядками матриці ($\mathrm{P}^T P$). Далі TF-IDF підхід може бути використано для зменшення впливу занадто популярних продуктів. Знаходяться найближчі елементи до елементів у історії користувача. Потрібно відсортувати ці елементи – продукти за значенням подібності, та вибрати перших k елементів. Це і є рекомендації.

3. РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОГО СЕРВІСУ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

3.1 Опис технологій, що використовуються

Вимоги до сучасних РС досить високі, адже об'єм даних, що має оброблюватися неймовірно великий, і, разом з тим, користувач не буде задоволений, якщо йому треба чекати кілька днів для отримання рекомендацій. Система має бути **масштабованою**, тобто, придатною для її використання з великими об'ємами даних. За рахунок використання багатьох індикаторів **якість рекомендацій**, які надає система значно підвищується. Ще одним критерієм є **швидке отримання рекомендацій.** Велика кількість запитів отримання рекомендацій надходить до РС на одиницю часу, усі запити мають обслуговуватися в режимі реального часу, щоб користувач мав позитивний досвід користування сервісом електронної комерції.

У даному розділі описується програмний продукт який являє собою РС що задовольняє вищевказаним потребам.

На ринку з'являється все більше і більше open-source технологій для вирішення задач збереження, обробки та пошуку даних. Це надає можливість не винаходити велосипед при вирішенні нової технічної задачі. Нові рішення потребують багато ресурсів та у випадку з невеликими компаніями не завжди будуть кращі за існуючі. Правильнішим підходом буде розбити таку задачу на підзадачі, та використовувати відповідні існуючі технології для вирішення підзадач де це можливо. Реалізацію РС можна розділити на декілька підсистем:

Система збору вхідних даних. Як було описано у розділі 1 РС має вхідний інтерфейс та вихідний інтерфейс. Ця система має реалізувати вхідний інтерфейс РС користувача, у нашому випадку вхідними даними являються індикатори подій користувача пов'язаних з певним продуктом. Система збору вхідних даних повинна мати інтерфейс для завантаження одиничних індикаторів подій та для великого набору таких даних. Система має швидко поглинати дані та надавати результат операції. Для вхідних інтерфейсів РС часто використовують пошукові движки, наприклад Elastic Search, або Apache Solr. Вони призначені для обробки великої

кількості запитів та гарно масштабуються. У проекті для збереження індикаторів взаємодії використовується ElasticSearch.

Система обробки даних (тренування моделі). Обробка даних полягає в перетворенні даних індикаторів взаємодії в рекомендації. Існують алгоритми які вирішують цю задачу в режимі реального часу, але на великих об'ємах даних такі алгоритми втрачають свою швидкість або точність. Переважно використовують спосіб тренування моделей офлайн. У проекті також використаний офлайн підхід тренування моделі. Для обчислень використовується *алгоритм спільного входження* з бібліотеки Apache Mahout, який читає вхідні дані з HDFS та виконується на розподіленій системі обчислень Apache Spark.

Система отримання рекомендацій. Результати алгоритму спільного входження записуються в ElasticSearch який слугує також і вихідним інтерфейсом PC користувача. Розглянемо основні сторонні технології, які використовуються у проекті.

Пошуковий движок Elastic Search

Elasticsearch - це швидкий, горизонтально масштабований і безкоштовний пошуковий движок (гібрид NoSQL бази даних наділеної багатою мовою запитів для неї). Користувач взаємодіє з сервісом через REST API. Elasticsearch сервіс представляє собою кластер, що складається з вузлів (нод). Дані зберігаються в індексах (які являються аналогом бази даних) та типах (які являються аналогом таблиць). Елементом даних є документ, який представляє собою JSON документ. Знаючи які пошукові запити даних будуть виконуватися користувач може задати правильний мапінг для кожного поля документа, що допоможе досягнути великої швидкості відповіді. Процес збереження документів у Elasticsearch називається індексацією. Кожен вузол Elasticsearch кластера бере участь у функціях індексації та приєднання до кластера, тобто вузол буде брати участь у певному пошуковому запиті шляхом пошуку даних, які він зберігає. Кожен вузол кластера може обробляти запити HTTP для клієнтів, які хочуть надіслати запит до кластера. Це робиться за допомогою HTTP REST API, який кластер викриває. Даний вузол потім отримує цей запит і несе

відповідальність за координування решти роботи. Крім того, даний вузол в рамках кластера знає про кожен вузол кластера і здатний пересилати запити до даного вузла за допомогою транспортного шару, тоді як рівень HTTP використовується виключно для спілкування з зовнішніми клієнтами. Всі вузли за замовчуванням приймають запити HTTP від клієнтів. Кожен вузол також може бути призначений як, так званий, основний вузол за замовчуванням. Головний вузол - це вузол, який відповідає за координацію змін у кластері, наприклад, додавання або видалення вузлів, створення чи видалення індексів тощо. Цей основний вузол оновлює стан кластера.[19]

Система розподілених обчислень Apache Spark

Apache Spark (від англ. Spark - іскра, спалах) - програмний каркас з відкритим вихідним кодом для реалізації розподіленої обробки неструктурованих і слабо структурованих даних, входить в екосистему проектів Hadoop. На відміну від класичного обробника з ядра Hadoop, що реалізує дворівневу концепцію MapReduce з дисковим сховищем, Spark використовує спеціалізовані примітиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значний виграш в швидкості роботи для деяких класів задач, зокрема, можливість багаторазового доступу до завантажених в пам'ять даних робить бібліотеку привабливою для алгоритмів машинного навчання. Проект надає програмні інтерфейси для мов Java, Scala, Python, R. Спочатку написаний на Scala, згодом додана істотна частина коду на Java для надання можливості написання програм безпосередньо на Java. Spark складається з ядра і кількох розширень, таких як Spark SQL (дозволяє виконувати SQL-запити над даними), Spark Streaming (надбудова для обробки потокових даних), Spark MLlib (набір бібліотек машинного навчання), GraphX (призначений для розподіленої обробки графів). Spark може працювати в середовищі кластера Hadoop під керуванням YARN та без компонентів ядра Hadoop, він підтримує кілька розподілених систем для зберігання даних - HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3. [20]

Розподілена файлова система HDFS

HDFS (Hadoop distributed file system) – розподілена файлова система, аналогічна файловій системі Linux, але фізично розгорнута на багатьох серверах. Ідея в тому, щоб можна було зберігати великі об'єми даних на машинах з відносно невеликими характеристиками. HDFS підтримує реплікацію даних. Архітектура системи складається з 3 видів демон процесів: NameNode, DataNode, SecondaryNameNode. [22] На рисунку 3.1 зображено архітектуру файлової системи HDFS[28].

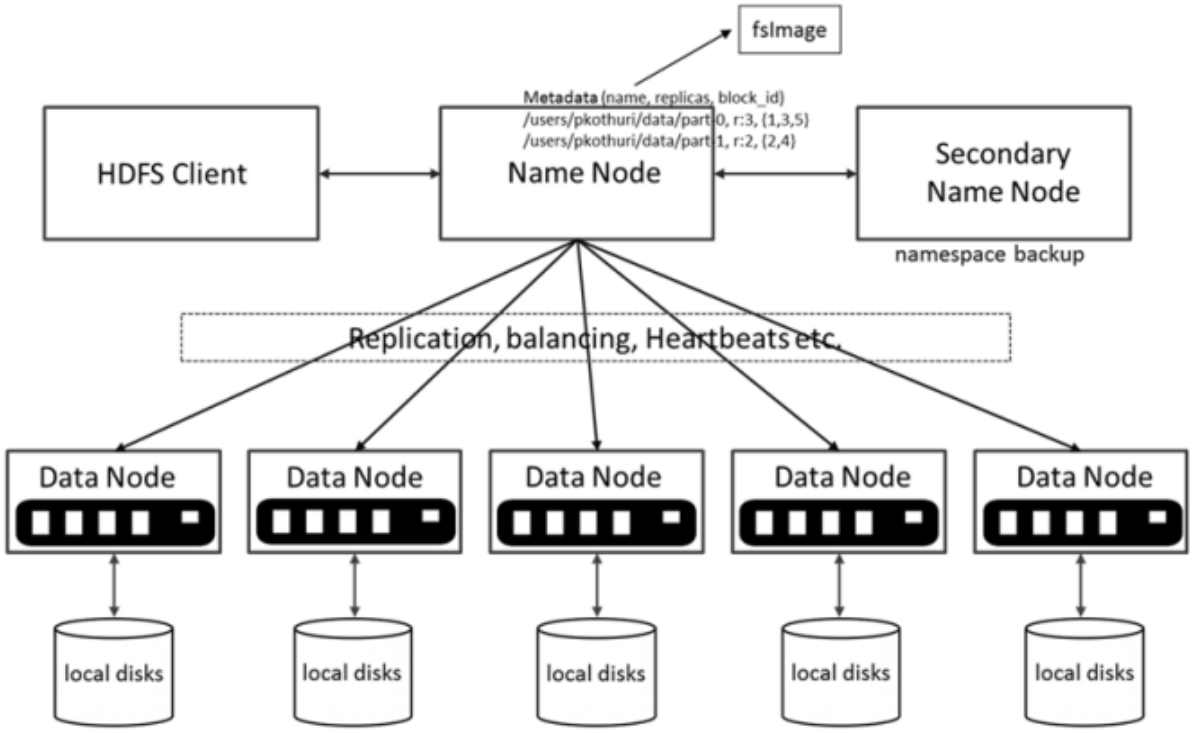


Рисунок. 3.1. Архітектура розподіленої файлової системи HDFS

Кожен файл даних в системі розбивається на блоки. За збереження блоків відповідає процес DataNode, а NameNode зберігає метадані про кожен файл з даними в системі: кількість та локації блоків кожного файлу даних. NameNode являється, так званим, single point of failure. Якщо NameNode втрачає метадані про файли з будь яких причин, файли з даними неможливо відновити. Для страхування існує процес SecondaryNameNode, який періодично копіює інформацію з NameNode і зберігає її. SecondaryNameNode може відновити стан NameNode. HDFS Client – це програма, яка відправлає запити по запису чи читанню даних в HDFS, вона звертається напряму до NameNode. [23]

Бібліотека Apache Mahout

Apache Mahout – бібліотека алгоритмів машинного навчання, які виконуються на системах розподілених обчислень таких як Hadoop MapReduce та Apache Spark. Mahoot вміщує ряд алгоритмів для реалізації рекомендаційних систем як на Hadoop так і на Spark. [24] Для Spark підтримується 2 алгоритма: spark-itemsimilarity та spark-rowsimilarity.

spark-itemsimilarity – алгоритм колаборативної фільтрації, який на основі неявних взаємодій користувачів з продуктами будує матрицю схожості продуктів між собою. Цікаво, що алгоритм підтримує 2 індикатори взаємодій: основний (наприклад, користувач придбав продукт), та другорядний (наприклад, користувач переглянув деталі продукту). Ідея в тому, що ми хочемо рекомендувати події, що позначаються основним індикатором (купівлю продуктів, наприклад), а події, на які вказує другий індикатор не завжди означають, що користувач позитивно відреагував на продукт. Наприклад, перегляд детального опису продукту не завжди означає, що користувач хоче придбати цей продукт. Алгоритм відфільтровує взаємодії з другого індикатора, які не корелюють з першим індикатором та будує дві матриці подібностей продуктів: за першим індикатором та за другим. Маючи ці дві матриці можна реалізувати якісну РС додавши бізнес логіку поверх отримання рекомендацій з матриць. Цей алгоритм використовується у проекті, описаному в даному розділі. [25]

spark-rowsimilarity – алгоритм фільтрації за контентом, який будує матрицю схожих продуктів на основі атрибутів кожного продукту. Підходить для створення рекомендаційних систем текстових документів, книг, музики та інших об’єктів, що не мають чітких атрибутів. [25]

Можливості та характеристики РС

Оскільки сучасні РС – це, безумовно, проекти у сфері Big Data технологій, то архітектура РС була спроектована з можливістю масштабування системи у майбутньому з мінімальними затратами.

Завдяки використанню пошукового движка ElasticSearch РС ми можемо отримати рекомендації в режимі реального часу.

Головною перевагою даної РС перед аналогами є можливість обробки багатьох типів індикаторів, що значно покращує якість рекомендацій. Кожен тип індикатор має пріоритет. Ідея в тому, що РС рекомендує події індикатора найвищого типу, дані індикаторів усіх інших типів фільтруються, залишивши тільки ті події, які мають високу кореляцію з подіями головного індикатора. Для кожного типу індикатора будується модель. Кожна модель може надавати рекомендації для певної підмножини користувачів. Список рекомендацій для користувача формується певним чином, описаним далі у цьому розділі.

3.2 Архітектура програмного продукту

Умовно архітектуру РС можна розділити на 2 компоненти: Online layer та Offline layer. Online layer надає можливість отримувати рекомендації з мінімальною затримкою через REST інтерфейс. [12]

3.2.1 Online рівень

Оскільки зі збільшенням даних збільшується і час тренування моделей, то тренувати нову модель після кожного нового рейтингу, зробленого користувачем, стає неможливим через величезне навантаження. Це потребує розподілення системи на real-time та batch рівні. Real-Time або онлайн рівень означає, що система надає рекомендації в online режимі. Матриця подібності продуктів, модель, створюється кожну добу, періодично, тобто офлайн. Архітектуру системи зображено в додатку В.

Система складається з 4 компонентів, помічених цифрами від 1 до 4 в кільці та інтегрованих сторонніх технологій, помічених цифрами 5, 6, 7 у кільці. Товстими стрілками помічено шлях пересування трансформованих вхідних даних системи. Тонкими стрілками позначено операції пов’язані з обробкою метаінформації про вхідні дані. Кожна стрілка позначена буквою англійського алфавіту від “a” до “i”. Система має мікросервісну архітектуру, що означає – кожен сервіс виконує свою бізнес-задачу.

Цифрою 1 позначено Менеджер Подій (Event Manager). Це REST сервіс, який слугує вхідним інтерфейсом РС. Цей сервіс спроектовано для обробки даних з

великою пропускною здатністю. Користувачі через REST API завантажують нові події (event) в систему. *Подія (event)* – це певна подія в часі, яка сталася на сайті електронної комерції і вказує на вираження інтересу певного рівня користувачем до продукту. Типові події на сайті електронної комерції:

- Користувач **u переглянув** інформацію про продукт **p**;
- Користувач **u додав до кошика** продукт **p**;
- Користувач **u купив** продукт **p**;

Подію можна описати терміном індикатор який був розглянутий у розділі 2. *Індикатором* у даному випадку являється кортеж **(u, type, p)**, де користувач з ідентифікатором **u** виразив відношення певного рівня **type** до продукту з ідентифікатором **p**. Event Manager індексує отримані події в в ElasticSearch, процес індексування зображено стрілкою з буквою *a*. Як було зазначено, система підтримує багато типів індикаторів подій. Перегляд інформації про товар, додавання товару до кошика та купівля товару – це все різні типи індикаторів, які означають різний рівень інтересу до продукту користувачем. Кожен тип індикатора має певний пріоритет. Індикатор з найвищим пріоритетом вважається типом індикатора, події якого система буде рекомендувати користувачам. Усі інші типи індикаторів вважаються другорядними і лише доповнюють опис поведінки користувача. Але, варто зазначити, що не всі події індикатора другорядного типу означають, що користувач виявляє інтерес до товару. Наприклад, якщо користувач переглянув інформацію про десятки предметів меблів, це не означає, що в усіх них він зацікавлений і, що він прагне рекомендацій схожих предметів. Можливо користувач просто шукає певний товар переглядаючи багато альтернатив. Саме тому система фільтрує події усіх індикаторів окрім першого з найвищим пріоритетом, залишивши тільки ті події, які мають високу кореляцію з подіями індикатора першого типу. Окрім завантаження подій Event Manager має інтерфейс для конфігурації пріоритету індикаторів. Ці метадані зберігаються в ElasticSearch, що зображено стрілкою з буквою *f*.

Цифрою 5 позначено пошуковий движок ElasticSearch – один із сторонніх

сервісів використаних для реалізації проекту. Головні задачі ElasticSearch: зберігати події індикаторів усіх типів, зберігати інформацію про типи індикаторів та їх пріоритет, зберігати моделі, треновані алгоритмом Mahout spark-itemsimilarity для подій типу кожного індикатора.

Цифрою 2 позначено Сервіс Рекомендацій (Recommender), який відповідає за вихідний інтерфейс PC. Цей сервіс спроектовано для обробки даних з великою пропускною здатністю. Сервіс представляє собою REST API для отримання рекомендацій. Стрілкою з буквою *g* позначено шлях, через який сервіс отримує метаінформацію про типи індикаторів. Стрілкою з буквою *e* позначено шлях через який Recommender виймає списки рекомендацій отримані від моделі кожного індикатора доти, доки це потрібно. Модель для кожного індикатора зберігається у ElasticSearch у своєму індексі яки вміщує ідентифікатор продукту, список рекомендованих цією моделлю ідентифікаторів продуктів для даного продукту та список ваг для таких рекомендацій. Обидва списки відсортовано в порядку спадання. Приклад:

```
    {
"objectId":"359844",
"recommendations":["463550","101075","333244"]
"preferences":[19.3165, 18.316588, 17.23454]
}
```

Алгоритм отримання рекомендацій можна описати наступним чином:

```
N // кількість рекомендацій, які треба отримати
user // користувач, для якого треба отримати рекомендації
indicators // список індикаторів відсортований за спаданням пріоритету
recommendations = [] // пустий список рекомендацій

k = 0
#1 Для індикатора indicators[k] отримати N рекомендацій для користувача user
        tmp = getRecommendations[indicators[k]].limit(N) // список рекомендацій від моделі індикатора k
recommendations += tmp
якщо рекомендацій для користувача менше ніж потрібно recommendations.size < N
k += 1
        якщо k == indicators.size =>
повернути рекомендації recommendations
інакше =>
N -= tmp.size
якщо T
перейти до #1
```

Рисунок 3.2 Алгоритм отримання рекомендацій

Для кожного типу індикатора у відсортованому за спаданням пріоритету списку індикаторів отримувати певну кількість рекомендацій, якщо у всіх моделях для індикаторів з вищим пріоритетом рекомендацій недостатньо. Такий простий алгоритм забезпечує досить високу ефективність, та високу швидкість.

3.2.2 Offline рівень

Асинхронність в комунікації між різними підсистемами великих систем допомагає досягти простоти в реалізації та гнучкості в проектуванні таких систем. ElasticSearch являється проміжною ланкою, що з'єднує online та offline рівні РС. Offline процеси асинхронно читають журнал подій користувача, або дані індикаторів користувачів та тренують моделі, які потім так само асинхронно записуються в ElasticSearch. Online рівень асинхронно записує дані індикаторів користувачів та асинхронно читає рекомендації з тренованих моделей.

Цифрою 3 позначено сервіс запуску тренування моделей індикаторів Job Runner. Цей сервіс представляє собою REST інтерфейс для запуску процесу тренування. Сторонній сервіс планувальник має відповідальність періодично відправляти запит на тренування моделі. Часто використовують період довжиною в 1 день, тобто моделі оновлюються один раз на добу. Це допомагає знизити навантаження на систему. Буквами *h* та *i* позначена взаємодія Job Runner сервісу з ElasticSerach та задачею тренування моделі, що буде розглянута далі. Ці взаємодії полягають у читанні метаінформації про індикатори та запуску процесу тренування, який приймає певні параметри, що залежать від індикатора.

Цифрою 4 позначено компонент системи, який реалізує алгоритм спільного входження, який був розглянутий раніше. Як реалізацію алгоритма було використано Mahout spark-itemsimilarity. spark-itemsimilarity утиліта являється Spark Application, тобто для її запуску потрібно середовище Apache Spark, яке зображено цифрою 7. Основними параметрами утиліти являються наступні:

- Шлях до вхідного файлу даних індикаторів поведінки користувача
- Шлях, куди записати файли тренованої моделі
- Назва основного індикатора (наприклад, "purchase")
- Назва другорядного індикатора (наприклад, "view")
- Spark master URL, параметр для запуску в середовищі Spark

Вхідними та вихідними шляхами являються директорії у файловій системі HDFS, яка

позначена цифрою 6.

Job Runner запускає компонент, позначений цифрою 4, який, в свою чергу, запускає виконання наступних задач:

1. Імпорт даних індикаторів користувачів у папку hdfs:///events. Цей процес виконується в середовищі Spark.
2. Запуск spark-itemsimilarity процесів рівній кількості другорядних індикаторів, або одного такого процесу, якщо другорядних індикаторів нема. Ці процеси виконуються паралельно в середовищі Spark. Результати записуються в директорію hdfs:///model-<ім'я індикатора>/. Результатом кожного такого процесу є 2 матриці: *similarity-matrix* та *cross-similarity-matrix*. Перша матриця представляє матрицю схожості продуктів отриманих методом колаборативної фільтрації від даних головного індикатора. Матриця *cross-similarity-matrix* представляє матрицю схожості продуктів отриману від даних другорядного індикатора, що корелюють між даними основного індикатора.
3. Імпорт матриць в ElasticSearch. Цей процес виконується в середовищі Spark.

Буквою *b* позначено копіювання даних індикаторів користувача в певний час до HDFS. Буквою *c* позначено імпорт цих даних до середовища Spark та їх обробка з допомогою Mahout spark-itemsimilarity. Буквою *d* позначено запис тренованих моделей – матриць до ElasticSearch.

Інтерфейс сервіса Event Manager

Вхідний інтерфейс PC складається з інтерфейсу для завантаження даних та інтерфейсу конфігурації типів індикаторів та їх пріоритетів.

Нижче зображено REST інтерфейс для конфігурації типів індикаторів та їх пріоритетів.

POST http://<host>:7777/api/model

```
Request:
{
  "primaryIndicator": "transaction",
  "secondaryIndicators": [
    {
      "name": "addtocart",
      "priority": 1
    },
    {
      "name": "view",
      "priority": 2
    }
  ]
}
Response:
None
```

Рисунок 3.3 API Event Manager

Поле primaryIndicaror вказує на назву основного індикатора. Поле secondaryIndicators вказує на список другорядних індикаторів, де name – ім'я індикатора, а priority – його пріоритет серед інших другорядних індикаторів. Менше значення поля priority означає вищий пріоритет.

Отримати конфігурацію індикаторів можна використовуючи наступне API:

POST http://<host>:7777/api/model

```
Request:
None
Response:
{
  "primaryIndicator": "transaction",
  "secondaryIndicators": [
    {
      "name": "addtocart",
      "priority": 1
    },
    {
      "name": "view",
      "priority": 2
    }
  ]
}
```

Рисунок 3.4 API конфігурації індикаторів

REST інтерфейс для завантаження даних індикаторів має вигляд:

POST http://<host>:7777/api/events

```
Request:
{
  "subjectId": "u1",
  "objectId": "i1",
  "indicator": "view"
}
Response:
None
```

Рисунок 3.5 API завантаження даних індикаторів

Де subjectId – ідентифікатор користувача, objectId – ідентифікатор продукта, indicator – ім’я індикатора події.

Інтерфейс сервіса Recommender

Рекомендації на сторінці продукту показують найкращу конверсію. Вихідний інтерфейс системи має REST API для отримання рекомендацій:

POST http://<host>:7778/api/recommendation/similarObjects

```
Request:
{
  "objectId": "i2",
  "size": 10
}
Response:
{
  "objectId": "i2",
  "recommendedObjectIds": [
    "i3",
    "i32",
    "i6",
    "i1",
    "i323",
    "i457",
    "i332",
    "i33",
    "i67"
  ]
}
```

Рисунок 3.6 API сервіса Recommender

У запиті поле objectId вказує на ідентифікатир продукта, для якого потрібно отримати схожі продукти в рекомендаціях, а поле size вказує на кількість таких рекомендацій. Результат вміщує 2 поля: objectId аналогічне полю objectId в запиті, поле recommendedObjectIds вміщує рекомендовані ідентифікатори продуктів.

Сервіс також дозволяє видалити усі рекомендації за домопогою наступного API:

DELETE http://<host>:7778/api/recommendation

Інтерфейс сервіса Job Runner

Для запуску тренування нової моделі система має окремий сервіс, цей сервіс розгорнутий на віртуальній машині, на якій запускаються bash скрипти для тренування моделі. При звертанні до наступного API Job Runner запускає скрипти, які в свою чергу запускають Spark Application:

POST http://<host>:7779/api/model/train

Рисунок 3.7 Job Runner API

У відповіді сервіс надає статус тренування моделі для кожного типу індикаторів, цей статут позначається у полі tasks, де name ім’я індикатора. Поле startedAt вказує на час в мілісекундах, коли тренування моделей було почато. Поле elapsed вказує на час в мілісекундах, який пройшов після запуску тренування моделі. Поле hasFailures вказує, чи була помилка тренування якоїсь із моделей. Якщо це поле true, то біля поля name з’являється поле message, де задається причина помилки. Тренування моделі неможливе, якщо уже запущено процес тренування. Коли тренування завершується, то у JSON статуса з’являється поле isFinished = true.

Асинхронно отримати статус тренування можна за допомогою наступного API. Цей підхід називається polling, коли клієнт періодично відправляє запит для перевірки статусу процесу.

GET http://<host>:7779/api/model/train

Рисунок 3.4 API статусу тренування

JSON у відповіді статусу має таку ж саму структуру, як і у відповіді API для тренування моделі.

3.3 Тестування з набором даних поведінки користувача на веб-сайті електронної комерції

Для тестування проекту було використано набір даних, який було зібрано з веб-сайту електронної комерції. Цей набір даних можна завантажити з веб-сайту Kaggle [11]

Набір даних складається з трьох файлів: файл із даними про дії користувача (events.csv), файл із властивостями продуктів з каталогу (itemproperties.csv) та файл, який описує дерево категорій (categorytree.csv). Дані були зібрані з реального веб-сайту електронної комерції. Це необроблені дані, тобто без будь-яких змістових перетворень, проте всі значення хешируються через конфіденційність.

Дані про поведінку, тобто такі події, як кліки, додавання у корзину, покупки, являють собою взаємодії, зібрані протягом 4,5 місяців. Відвідувач може здійснити три типи подій, а саме: «перегляд», «додаток» або «транзакція» (покупка). Загалом є

2 756 101 події, включаючи 2 664 312 перегляди, 69 332 додати до візків та 22 457 транзакцій, здійснених 1 407 580 унікальними відвідувачами. Приблизно 90% властивостей подій можна знайти у файлі «item_properties.csv».

Наприклад:

"1439694000000,1, view, 100", означає visitorId = 1, переглянув елемент з id = 100 о 1439694000000 (часова мітка Unix)

"1439694000000,2, транзакція, 1000,234" означає, що відвідувач з Id = 2 придбав товар з id = 1000 в транзакції з id = 234 о 1439694000000 (часова мітка Unix)

Файл із властивостями продуктів (item_properties.csv) включає 20 275 902 рядків, тобто різні властивості, описуючи 417 053 унікальних елементів. Файл розділений на 2 файли через обмеження розміру файлу. Оскільки властивість елемента може змінюватись у часі (наприклад, ціна змінюється з часом), кожен рядок у файлі має відповідну часову позначку. Іншими словами, файл складається з об'єднаних знімків на кожен тиждень у файлі з даними про поведінку. Однак якщо властивість елемента є постійною протягом спостережуваного періоду, у файлі буде присутній лише один момент знімка.Наприклад, у нас є три властивості для одного елемента та 4 знімки на тиждень, як нижче:

timestamp,itemid,property,value

1439694000000,1,100,1000

1439695000000,1,100,1000

1439696000000,1,100,1000

1439697000000,1,100,1000

1439694000000,1,200,1000

1439695000000,1,200,1100

1439696000000,1,200,1200

1439697000000,1,200,1300

1439694000000,1,300,1000

1439695000000,1,300,1000

1439696000000,1,300,1100

1439697000000,1,300,1100

Після злиття знімка це виглядає так:

1439694000000,1,100,1000

1439694000000,1,200,1000

1439695000000,1,200,1100

1439696000000,1,200,1200

1439697000000,1,200,1300

1439694000000,1,300,1000

1439696000000,1,300,1100

Оскільки властивість = 100 є постійною протягом часу, властивість = 200 має різні значення для всіх знімків, властивість = 300 було змінено один раз.Файл властивостей елемента містить стовпчик часових позначок, оскільки всі вони залежать від часу, та властивості можуть змінюватися з часом, наприклад ціна, категорія тощо. Спочатку цей файл складався із знімків кожного тижня у файлі подій і містив понад 200 мільйонів рядків. Таким чином, постійні значення відображатимуться лише один раз у файлі. Ця дія зменшила кількість рядків у 10 разів. Усі значення у файлі "item_properties.csv", виключаючи властивості "categoryid" та "available", були хешировані. Значення властивості "categoryid" містить ідентифікатор категорії предмета. Значення властивості "available" містить доступність елемента, тобто 1 означає, що елемент був доступний, інакше 0.

Усі числові значення на початку були позначені знаком "n" і мають 3-х точну точність після десяткових знаків, наприклад, " 5 "стане" n5.000 "," -3.67584 "стане" n-3.675 ". Усі слова в текстових значеннях були нормалізовані (процедура Stemming [10]) та хешировані, цифри оброблялися як вище, напр. текст "Hello world 2020!" стане "24214 44214 n2020.000"

Файл ієрархії категорій має 1669 рядків. Кожен рядок у файлі вказує дочірню categoryId та відповідну батьківську. Наприклад: Рядок "100,200" означає, що categoryId = 100 належить categoryId = 200 Рядок "300" означає, що categoryid не має

батьківського вузла. У додатку А можна переглянути проаналізовані дані: товари, що були найчастіше переглянуті або придбані.

Проаналізуємо найбільш популярні товари за різними показниками: покупка, додавання у корзину, перегляд (Рисунок 3.9, 3.10, 3.11)

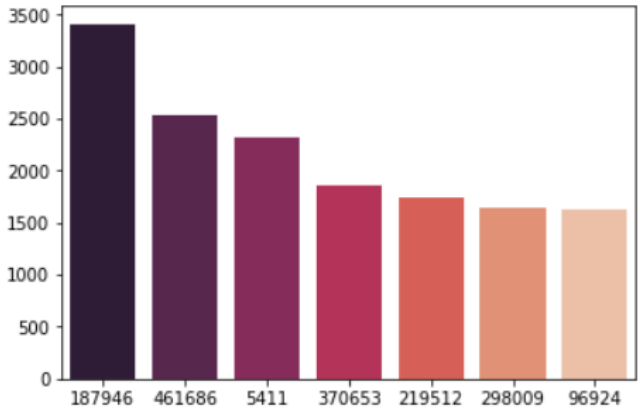


Рисунок 3.9 Позиції, що були найчастіше переглянуті. Вісь x – унікальний ідентифікатор продуку, вісь y – кількість переглядів.

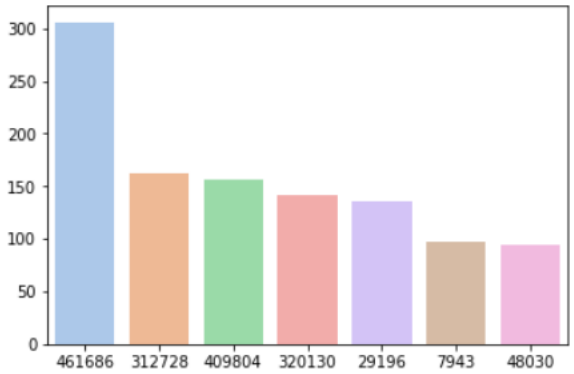


Рисунок 3.10 Позиції, що були найчастіше додані у кошик. Вісь x – унікальний ідентифікатор продуку, вісь y – кількість додавань у корзину.

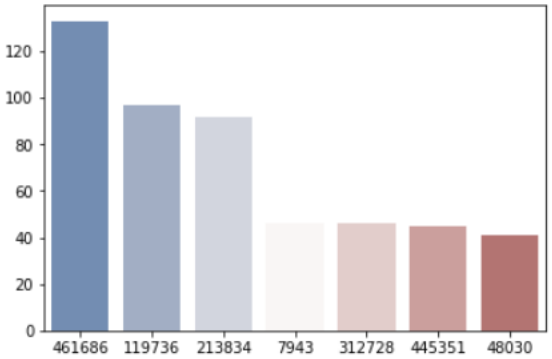


Рисунок 3.11 Позиції, що були найчастіше куплені. Вісь x – унікальний ідентифікатор продуку, вісь y – кількість транзакцій.

Для тестування системи було використано файл events.csv. Поле transactionid було ігноровано. Розмір файла events.csv складає 32.88 MB. Кількість подій купівлі, додавання до кошика та перегляду інформації про товар складає 2 756 101. [11]

Усі ці події було завантажено в систему через Events Service batch API, описане у попередньому пункті. Для завантаження набору даних використовуються скрипт додатку А. Завантаження набору даних займає 3 хвилини. Тренування моделі займає 7 хвилин.

Отримуємо 7 рекомендацій для продукту з id 14532:

POST http://localhost:7778/api/recommendation/similarObjects

Request: {

```
“objectId”: “14532”
“size”: 7
}
```

Response: {

```
“objectId”: “14532”
“recommendedObjectIds: [
    “87453”,
    “64562”,
    “901462”,
    “52802”,
    “632356”,
    “2452”,
    “632”
]
```

Для перевірки точності РС найкраще підходить A/B тестування, що не входить в рамки даного дипломного проекту.

ВИСНОВКИ

Рекомендаційні системи є системами фільтрації інформації. У сучасні дні складно уявити своє життя без таких помічників.

У першому розділі було розглянуто основні поняття теорії рекомендаційних систем та функції, які вони виконують. Були визначено типи даних з якими працюють РС: користувачі, продукти, рейтинги. Було описано основні алгоритми, та підходи до обчислення рекомендацій. Було порівняно сучасні методи до обчислення рекомендацій та визначено переваги і недоліки кожного з них.

Було поглиблено розглянуто підходи до побудови РС. Для того, щоб покрити недоліки одного алгоритму перевагами іншого використовують гібридні підходи до обчислення рекомендацій. Також, було визначено точність РС та основні її метрики, такі як: precision, recall та AUC. Для оцінки точності РС потрібно мати механізм для оцінки похибки її результатів. У цьому розділі було запропоновано новий підхід до створення рекомендаційних систем у сфері електронної комерції, який використовує алгоритм ССО.

Реалізація включила у себе проектування системи мікросервісів, які отримують та зберігають дані, виконують запити на тренування моделі та надають готові рекомендації.

Метриками оцінки якості рекомендацій є порівняння продажів певних товарів до та після запуску системи, визначення відсотку конверсії на запропонованому товарі, відсоток покупок таких продуктів, тощо. Оскільки, особливість рекомендаційної системи саме у тому, що вона використовує різні показники та «неявні» відгуки для визначення найкращого товару для певної особи, повністю оцінити її якість можливо лише при тестуванні на досвіді реальних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recommender systems introduction. [Електронний ресурс] - Режим доступу до ресурсу: https://www.coursera.org/learn/recommender-systems-introduction/home/week/1
2. Aggarwal C. Recommender Systems / Charu Aggarwal., 2016. – 498 с.
3. Введение в рекомендательные системы [Електронний ресурс] - Режим доступу до ресурсу https://habr.com/ru/post/476222/
4. 10 Product Recommendation Techniques to Improve UX and Conversions [Електронний ресурс] - Режим доступу до ресурсу: https://conversionxl.com/blog/product-recommendations/
5. The Netflix Prize Rules. [Електронний ресурс] - Режим доступу до ресурсу: http://www.netflixprize.com/rules.html
6. Recommender Systems: An Introduction / C.Dietmar Jannach, M. Zanker, A. Felfernig, A. Friedrich. – 2010. – №2. – С. 315.
7. Forbes - How Much Data Do We Create Every Day [Електроний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#3f7a3fe460ba
8. Hybrid Recommender Systems: Survey and Experiments. [Електронний ресурс] - Режим доступу до ресурсу: https://pdfs.semanticscholar.org/5880/b9bc3f75f4649b8ec819c3f983a14fca9927.pdf
9. Approaches, Issues and Challenges in Recommender Systems: A Systematic Review [Електронний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/312507062_Approaches_Issues_and_Challenges_in_Recommender_Systems_A_Systematic_Review
10. Stemming [Електроний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Stemming

11. Retailrocket recommender system dataset [Електронниний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/retailrocket/ecommerce-dataset#events.csv

12. Andrew Havens. Beginners guide to creating a REST API [Електронний ресурс]. – Режим доступу: http://www.andrewhavens.com/posts/20/beginners-guide-to-creating-a-rest-api/

13. Янг Б. Объектно-ориентированный анализ и проектирование с примерами приложений. 3-е издание [Текст, електронний ресурс] / Бобби Янг, Джим Коналлен, Гради Буч. – М.: Вильямс, 2008. – 720 с.

14. Метрики рекомендательной системы Imhonet.ru. [Електронний ресурс] - Режим доступу до ресурсу: https://habrahabr.ru/company/dca/blog/281066/

15. McLeod D. Collaborative Filtering for Information Recommendation Systems [Електронний ресурс] / D. McLeod, A. Y. Chen // Research Reports. – 2009. – Режим доступу до ресурсу: http://research.create.usc.edu/cgi/viewcontent.cgi?article=1101&context=nonpublished_reports.

16. Koren Y. Matrix Factorization Techniques for Recommender Systems / Y. Koren, B. Robert, V. Chris. // Journal Computer. – 2009. – №42 (8). – pp. 30–37.

17. Dunning T. Practical Machine Learning: Innovations in Recommendation / T. Dunning, E. Friedman, A. Felfernig // O'Reilly Media, Inc. , 2014. – 56 с.

18. Lucene documentation [Електронний ресурс] – Режим доступу до ресурсу: https://lucene.apache.org/core/7_5_0/index.html

19. Elasticsearch documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/index.html

20. Elasticsearch Spark support [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/en/elasticsearch/hadoop/current/spark.html

21. Apache Spark Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://spark.apache.org/docs/2.2.1/
22. HDFS User Guide [Електронний ресурс] – Режим доступу до ресурсу: https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html
23. Hadoop tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/hadoop/
24. Mahout documentation [Електронний ресурс] – Режим доступу до ресурсу: https://mahout.apache.org/
25. Building a Correlated Cross-Occurrence (CCO) Recommenders with the Mahout CLI [Електронниний ресурс] – Режим доступу до ресурсу: http://mahout.apache.org/users/recommender/intro-cooccurrence-spark.html
26. Amazon [Електронний ресурс] – Режим доступу до ресурсу: https://www.amazon.com/
27. Rozetka.ua [Електронний ресурс] – Режим доступу до ресурсу: https://rozetka.com.ua/
28. An introduction to HDFS [Електронний ресурс] – Режим доступу до ресурсу: https://dzone.com/articles/an-introduction-to-hdfs

Додаток А. Проаналізовані тестові дані

```
import numpy as np # лінійна алгебра
import pandas as pd # обробка даних, CSV I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

# Дані знаходяться у директорії "../input/" .

import os
print(os.listdir("../input"))
```

```
['item_properties_part1.csv', 'events.csv', 'item_properties_part2.csv', 'category_tree.csv']
```

```
items1 = pd.read_csv('../input/item_properties_part1.csv')
items2 = pd.read_csv('../input/item_properties_part2.csv')
items = pd.concat([items1, items2])
items.head(10)
```

	timestamp	itemid	property	value
0	1435460400000	460429	categoryid	1338
1	1441508400000	206783	888	1116713 960601 n277.200
2	1439089200000	395014	400	n552.000 639502 n720.000 424566
3	1431226800000	59481	790	n15360.000
4	1431831600000	156781	917	828513
5	1436065200000	285026	available	0
6	1434250800000	89534	213	1121373
7	1431831600000	264312	6	319724
8	1433646000000	229370	202	1330310
9	1434250800000	98113	451	1141052 n48.000

Розмір даних

```
items.shape
```

```
(20275902, 4)
```

+ Code + Markdown

```
import datetime
times =[]
for i in items['timestamp']:
    times.append(datetime.datetime.fromtimestamp(i//1000.0))
```

```
items['timestamp'] = times
```

```
items.head(10)
```

	timestamp	itemid	property	value
0	2015-06-28 03:00:00	460429	categoryid	1338
1	2015-09-06 03:00:00	206783	888	1116713 960601 n277.200
2	2015-08-09 03:00:00	395014	400	n552.000 639502 n720.000 424566
3	2015-05-10 03:00:00	59481	790	n15360.000
4	2015-05-17 03:00:00	156781	917	828513
5	2015-07-05 03:00:00	285026	available	0
6	2015-06-14 03:00:00	89534	213	1121373
7	2015-05-17 03:00:00	264312	6	319724
8	2015-06-07 03:00:00	229370	202	1330310
9	2015-06-14 03:00:00	98113	451	1141052 n48.000

```
# Завантажуємо dataset дій користувача

events = pd.read_csv('../input/events.csv')
```

+ Code + Markdown

```
events.head(10)
```

	timestamp	visitorid	event	itemid	transactionid
0	1433221332117	257597	view	355908	NaN
1	1433224214164	992329	view	248676	NaN
2	1433221999827	111016	view	318965	NaN
3	1433221955914	483717	view	253185	NaN
4	1433221337106	951259	view	367447	NaN
5	1433224086234	972639	view	22556	NaN
6	1433221923240	810725	view	443030	NaN
7	1433223291897	794181	view	439202	NaN
8	1433220899221	824915	view	428805	NaN
9	1433221204592	339335	view	82389	NaN

```
events.shape
```

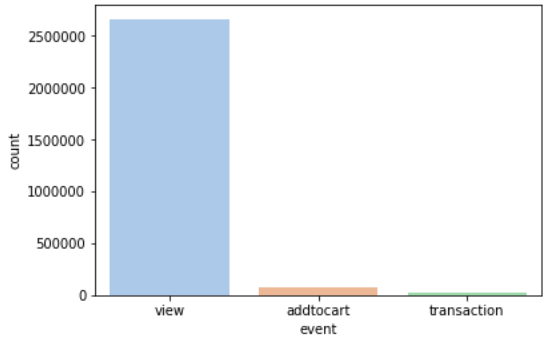
```
(2756101, 5)
```

Проаналізуємо види дій користувача

```
print(events['event'].value_counts())
sns.countplot(x= 'event', data=events, palette="pastel")
```

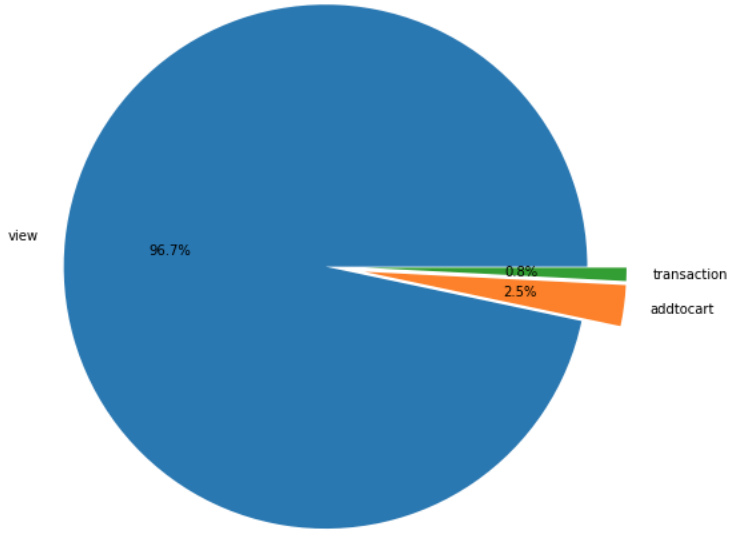
```
view           2664312
addtocart        69332
transaction      22457
Name: event, dtype: int64
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdedac95e48>
```




```
data = events.event.value_counts()
labels = data.index
sizes = data.values
explode = (0, 0.15, 0.15)  # explode 1st slice
plt.subplots(figsize=(8,8))
# Plot
plt.pie(sizes, explode=explode, labels=labels,autopct='%1.1f%%', shadow=False, startangle=0)

plt.axis('equal')
plt.show()
```



+ Code + Markdown

Дерево категорій

:
```
category_tree = pd.read_csv('../input/category_tree.csv')
```

:
```
category_tree.head(10)
```

3]:

	categoryid	parentid
0	1016	213.0
1	809	169.0
2	570	9.0
3	1691	885.0
4	536	1691.0
5	231	NaN
6	542	378.0
7	1146	542.0
8	1140	542.0
9	1479	1537.0

Category IDs показує звязок різних продуктів між собою, наприклад Category ID 1016 дочірній продукт ID 213.

Кількість продуктів з ID 1016

```
items.loc[(items.property=='categoryid')&(items.value == '1016')].sort_values('timestamp').head()
```

4]:

	timestamp	itemid	property	value
7236969	2015-05-10 03:00:00	6777	categoryid	1016
8597591	2015-05-10 03:00:00	161686	categoryid	1016
9496408	2015-05-10 03:00:00	276491	categoryid	1016
6880131	2015-05-10 03:00:00	443058	categoryid	1016
7202531	2015-05-10 03:00:00	462004	categoryid	1016

Кількість унікальних відвідувачів та сума загалом

```
# all unique visitors
all_customers = events['visitorid'].unique()
print("Unique visitors:", all_customers.size)

# all visitors
print('Total visitors:', events['visitorid'].size)
```

```
Unique visitors: 1407580
Total visitors: 2756101
```

+ Code + Markdown

Кількість користувачів, які щось придбали

```
customer_purchased = events[events.transactionid.notnull()].visitorid.unique()
customer_purchased.size
```

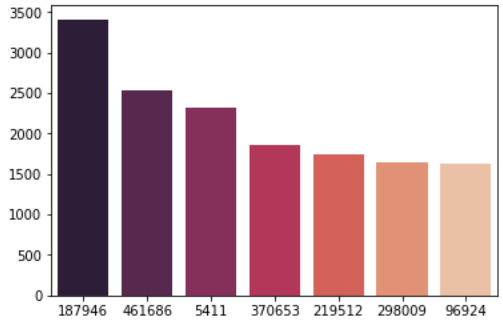
6] 11719

Існує 1407580 унікальних відвідувачів, 11719 відвідувачів зробили хоча б одну покупку

Позиції, які мають найбільше переглядів

```
import operator
grouped = events.groupby('event')['itemid'].apply(list)
views = grouped['view']
# creating dictionary for key value pair
count_view ={}
#since views is a list, we will convert it into numpy array for further manipulations
views = np.array(views[:])
#counting uniques values of views of this numpy views array
unique, counts = np.unique(views, return_counts=True)
# converting unique and counts as a dictionay with key as unique and value as counts
count_view = dict(zip(unique, counts))
#sorting the dictionary
sort_count_view = sorted(count_view.items(), key = operator.itemgetter(1), reverse = True)
# keeping number of unique views on X-axis
x = [i[0] for i in sort_count_view[:7]]
# keeping count number of views on Y-axis
y = [i[1] for i in sort_count_view[:7]]
sns.barplot(x, y, order=x, palette="rocket")
```

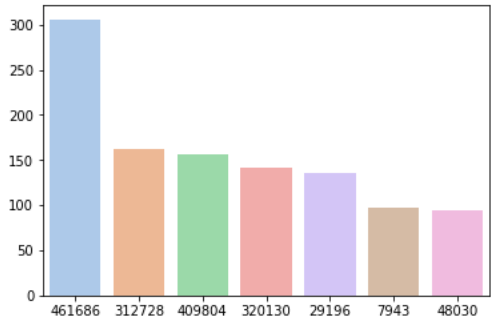
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf15c460b8>
```



Позиції, які були найчастіше додані до кошика

```
addtocart = grouped['addtocart']
# creating dictionary for key value pair
count_addtocart ={}
#since addtocart is a list, we will convert it into numpy array for further manipulations
addtocart = np.array(addtocart[:])
#counting uniques values of addtocart items of this numpy addtocart array
unique, counts = np.unique(addtocart, return_counts=True)
# converting unique and counts as a dictionay with key as unique and value as counts
count_addtocart = dict(zip(unique, counts))
#sorting the dictionary
sort_count_addtocart = sorted(count_addtocart.items(), key = operator.itemgetter(1), reverse = True)
# keeping number of unique views on X-axis
x = [i[0] for i in sort_count_addtocart[:7]]
# keeping count number of views on Y-axis
y = [i[1] for i in sort_count_addtocart[:7]]
sns.barplot(x, y, order=x, palette="pastel")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdfb733dcf8>
```

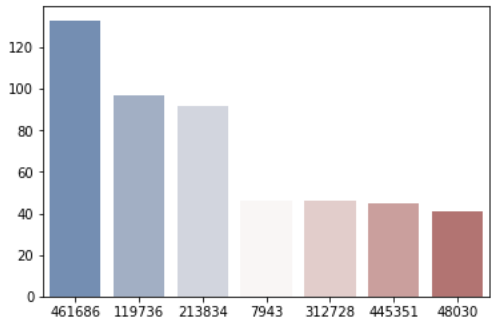


+ Code + Markdown

Позиції, які були найчастіше куплені

```
transaction = grouped['transaction']
# creating dictionary for key value pair
count_transaction ={}
#since addtocart is a list, we will convert it into numpy array for further manipulations
transaction = np.array(transaction[:])
#counting uniques values of addtocart items of this numpy addtocart array
unique, counts = np.unique(transaction, return_counts=True)
# converting unique and counts as a dictionay with key as unique and value as counts
count_transaction = dict(zip(unique, counts))
#sorting the dictionary
sort_count_transaction = sorted(count_transaction.items(), key = operator.itemgetter(1), reverse = True)
# keeping number of unique views on X-axis
x = [i[0] for i in sort_count_transaction[:7]]
# keeping count number of views on Y-axis
y = [i[1] for i in sort_count_transaction[:7]]
sns.barplot(x, y, order=x, palette="vlag")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf1572b588>
```



+ Code + Markdown

Як один із пунктів гібридної рекомендації, можемо запропонувати користувачу список товарів, які інші користувачі пр
продуктом, який він наразі дивиться

```
# список користувачів, що зробили покупку
customer_purchased = events[events.transactionid.notnull()].visitorid.unique()

#список покупок усіх користуваців
purchased_items = []

for customer in customer_purchased:
    purchased_items.append(list(events.loc[(events.visitorid == customer) & (events.transactionid.notnul
```

+ Code + Markdown

```
purchased_items[:7]
```

```
[[356475],
 [15335,
  380775,
  237753,
  317178,
  12836,
  400969,
  105792,
  25353,
  200793,
  80582,
  302422],
 [81345],
 [150318, 49521],
 [310791, 299044],
 [54058,
  284871,
```

Визначимо функцію, що покаже продукти, які були придбані разом один и тим самим покупцем

```
def recommend_items(item_id, purchased_items):
    recommendation_list =[]
    for x in purchased_items:
        if item_id in x:
            recommendation_list +=x

    # remove the pass item from the list and merge the above created list
    recommendation_list = list(set(recommendation_list) - set([item_id]))
    return recommendation_list
```

Зробимо рекомендацію базуючись на данних попереднього покупця, коли користувач переглядає позицію з item_id =

```
recommend_items(200793, purchased_items)
```

```
[105792, 12836, 80582, 380775, 15335, 400969, 25353, 302422, 237753, 317178]
```

```
events.head(5)
```

+ Code + Markdown

Додаток В. Архітектура системи