Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Реалізація алгоритмів рекомендаційних систем

Курсова робота

за спеціальністю „Інженерія програмного забезпечення” 6.050103

Керівник курсової роботи
кандидат технічних наук
Ковалюк Тетяна Володимирівна

Виконав студент

Безштанько В. В.

Київ – 2020

Тема: Реалізація алгоритмів рекомендаційних систем

Календарний план виконання роботи:

№ п/п	**Назва етапу контрольної роботи**	**Термін виконання етапу**	**Примітка**
1.	Отримання завдання на курсову роботу	11.10.2019	
2.	Огляд літератури за темою роботи	5.03.2020	
3.	Написання текстової частини	16.04.2020	
4.	Написання програмної частини	23.04.2020	
5.	Надання роботи керівнику для перевірки	30.04.2020	
6.	Коригування за результатами перевірки	9.05.2020	
7.	Захист курсової роботи	18.05.2020	

Студент _____Безштанько_В._В._____
Керівник _____Ковалюк_Т._В._____

Зміст

Вступ

Рекомендаційні системи – відносно новий напрямок розвитку інформаційних технологій, що виник з появою сучасних інтернет ресурсів, орієнтованих на користувача. Серед систем, для яких рекомендаційні алгоритми грають ключову роль, потрібно звернути особливу увагу на соціальні мережі, онлайн кінотеатри, інтернет-магазини, засоби контекстної реклами, тощо. Розробка та впровадження алгоритмів надання рекомендацій може суттєво збільшити прибутки інформаційних систем такого характеру, що було не раз доведено на практиці. Так, наприклад, 2/3 всіх фільмів, що було переглянуто користувачами Netflix, було знайдено саме завдяки алгоритмам рекомендацій, а продажі товарів, рекомендованих алгоритмами Amazon, складають приблизно 35% всього доходу. Тож рекомендаційні системи входять до сучасних засобів клієнтоорієнтованих систем, і складають важливу частину функціоналу таких систем.

Разом з тим, алгоритми рекомендаційних систем можуть бути застосовані для дослідження зв’язків певних явищ, та надавати рекомендації щодо покращення взаємодії користувачів. За приклад можна навести випадок з банківської сфери, коли було виявлено зв’язок між банкоматами, де користувачі знімали гроші, та відповідними магазинами, де вони їх витрачали, що дозволило повідомляти про наявні акційні пропозиції відповідних торгових мереж.

Зважаючи на таку визначну користь рекомендаційних систем у наведених вище областях, предметом курсової роботи було обрано дослідження алгоритмів рекомендаційних систем та програмну реалізацію частини таких алгоритмів.

Вже зараз суттєво відмітити, що кожен розглянутий у роботі підхід має свої переваги і недоліки, та, в першу чергу, залежить від конкретної предметної області інформаційної системи, для якої впроваджують рекомендаційні алгоритми, вимог до функціонування такої системи, та наявної інформації про користувачів та послуги цієї інформаційної системи. Тому не існує універсального механізму вибору застосовуваних алгоритмів рекомендаційних

систем, і кожен окремий випадок повинно бути розглянуто з точки зору відповідності наявних ресурсів до очікуваних вимог.

Теоретична частина

Фактори вибору алгоритмів рекомендаційних систем

Не зважаючи на відносно недавній час виникнення потреби та інтересу широкого користувача до рекомендаційних систем, різноманіття використовуваних рішень надзвичайно широке. Це значною мірою спричинено як відсутністю однозначно сформованих підходів до рішення задачі надання рекомендацій інформаційними системами, так і принциповою неможливістю створення єдиного підходу до задач такого типу, адже вибір оптимального розв'язку має критичну залежність від: даних, що ними маніпулює система, їх структурованості, характерної природи та стрімкості перетворення у дійсному інформаційному середовищі з часом. Зважаючи на сказане, важливо розуміти порядок оцінки рекомендаційних систем для вибору оптимальних рішень щодо їх подальшої розробки та впровадження.

Найпершим елементом при виборі рекомендаційної системи і, відповідно, закладуваних у її основі алгоритмів, має бути *оцінка використовуваного профілю користувача*. Важливо відповісти на наступні запитання: чи повинна система зберігати інформацію щодо місця проживання користувача, його рівня освіти, статі, віку, бажаних засобів оплати, чи будь якої іншої персональної інформації. Кожен елемент профілю користувача може бути вагомим, у залежності від інформаційної моделі. Безпосередньо пов'язаною з цією задачею, є також задача генерації початкового користувацького профілю, метою якої є якнайшвидше наповнення інформації про користувача, для забезпечення користувача можливістю повноцінно використовувати надавані рекомендаційними системами можливості. Найзручнішим механізмом забезпечення швидкої інтеграції користувача беззаперечно є соціальні мережі, тому інтеграція засобів аналізу соціальних мереж може бути гарним фундаментом при розробці інформаційних систем, що мають на меті впроваджувати рекомендаційні системи для підвищення ефективності своєї роботи.

Наступною важливою рисою рекомендаційних систем є їх можливість чи неможливість забезпечення зворотного зв’язку з користувачем. Більшість сучасних систем покладаються саме на цей елемент інформаційних систем, намагаючись явно чи неявно оцінити реакцію користувача на ті, чи інші результати. Відповідно інтеграція таких засобів у системи дозволяє звертатись до більш персоніфікованих алгоритмів надання рекомендацій.

Де-які системи звертають також увагу на можливість пристосування до змін смаків користувача, які неминуче виникають з часом. Можливість системи «забувати» давні запити може суттєво покращити користувацький досвід у проектах, орієнтованих на довгострокове використання.

Зауваження схожого характеру стосуються також і *об’єктного наповнення інформаційних систем*. Застосування розширених, добре структурованих даних щодо об’єктів, якими оперує система, може стати важливим елементом алгоритмів рекомендацій, і вагомо вплинути на якість цих алгоритмів.

Наявність згаданих можливостей та даних у інформаційній системі може відкрити шлях до впровадження таких алгоритмів як *демографічна фільтрація*(demographic filtering), *фільтрація, що спирається на контент* (content-based filtering), та *колаборативна фільтрація*(collaborative filtering).

Демографічна фільтрація базується на аналізі профілю користувача. Використовуючи особисту інформацію для аналізу і співставлення з іншими користувачами, алгоритми демографічної фільтрації зосереджують свою увагу переважно на пошуку загальних рекомендацій для користувачів певних груп.

Алгоритми колаборативної фільтрації являють собою засоби аналізу матриці зв’язків користувачів та послуг – об’єктів інформаційної системи для подальшого надання рекомендацій, засновуючись на цих зв’язках.

Фільтрація, що спирається на контент, з одно боку є найбільш персоніфікованим варіантом пошуку, проте складнощі впровадження і підтримки систем такого характеру часто нівелюють їх користь. Цей вид

фільтрації може застосовуватись як спираючись на дані профілю користувача, так і на дані об’єктів інформаційної системи. Для впровадження систем такого характеру можна, наприклад, звернутись до такого розповсюдженого алгоритму інформаційного пошуку як tf-idf, якщо об’єкти інформаційної системи можливо представити файлами з якимось змістовним описом, тощо.

Найвизначніші алгоритми рекомендаційних систем

Найбільш застосовними алгоритмами рекомендаційних систем у природі є алгоритми колаборативної фільтрації, та фільтрації, що спирається на контент. Не зважаючи на їх безумовну користь, вони також мають і свої недоліки.

Алгоритми колаборативної фільтрації переважно спрямовані на збір інформації щодо відгуків користувачів стосовно взаємодії самих користувачів та надаваних інформаційною системою послуг. Алгоритми колаборативної фільтрації працюють, спираючись на user-item матрицю, комірки якої заповнюють опосередковано у процесі взаємодії з інформаційною системою самі користувачі. Проте, на практиці, користувачі дуже рідко бажають самостійно заповнювати таку інформацію, системою відгуків користується в найкращому випадку 15% користувачів, тому така матриця дуже розріджена. Алгоритми колаборативної фільтрації мають, в тім, великий недолік – так звана проблема «холодного старту»: нові користувачі, або нові об’єкти, додавані до системи, не надають про себе зазвичай жодної інформації, тому можуть бути загублені у інформаційному потоці.

Алгоритми фільтрації, засновані на контенті, з іншого боку, мають ставити ще більше запитів до користувача, якщо бажають ефективно використовувати профільні дані користувача. На практиці таке навантаження на користувача може навіть відлякнути його. Однак, користувачі, що створюють контент, додаючи до інформаційної системи, більш схильні надавати додаткову інформацію, адже це сприяє ефективності рекомендацій таких засобів користувачам. Вдало надані користувачами дані, представлені у інформаційних системах, що використовують алгоритми фільтрації, засновані на контенті, зазвичай

складають широку ґрунтовну базу для роботи цих алгоритмів Тому гарні системи, засновані на фільтрації контенту, значно менше підпадають під проблему «холодного старту».

У сучасних системах, на практиці, зазвичай застосовують комбінації цих та інших підходів, характер їх поєднання може бути надзвичайно різним. З точки зору алгоритмів, найбільшу поширеність все ж на сьогодні завоював алгоритм матричної факторизації, переможець конкурсу Netflix prize. Це один з алгоритмів колаборативної фільтрації, заснований на формальній моделі розкладу матриць на менші матриці для зменшення кількості обрахунків. Загальною характеристикою алгоритмів, заснованих на моделях, можна вважати такий важливий елемент, як закладена у них здатність до знаходження рішень незалежно від людини, що може привести до цілком неочевидних, але не менш ефективних рішень.

У зв'язку з нерідко виявлюваною схильністю рекомендаційних систем до «перенавчання», значним фактором у розробці сучасних рекомендаційних систем є так звана властивість «серендипності» - можливості надання рекомендацій, неочевидно для людини пов'язаних з минулими інтересами користувача, проте таких, що виводять його з кола постійних повторень однакових рекомендацій. З іншого боку, високі навантаження на обчислювальні ресурси найбільших інформаційно-орієнтованих компаній, теж створює проблеми для використання описаних алгоритмів. Тому найпотужніші інформаційні титани спрямовують ресурси на впровадження більш нових методів впровадження рекомендаційних систем, у тому числі у машинне навчання. Проте зазначені у цьому розділі методи досі широко вживані в інформаційних системах з більш традиційними вимогами.

Алгоритми колаборативної фільтрації

Алгоритми колаборативної фільтрації, базуючись на матриці user-item, можуть мати під своєю основою або алгоритми засновані на формальному підході(як алгоритми матричної факторизації), або алгоритми рекомендацій за

подібністю. Серед алгоритмів рекомендацій за подібністю найбільш відомими є user-user та item-item алгоритми колаборативної фільтрації.

USER-USER алгоритми

Алгоритми цієї категорії використовують user-item матрицю для пошуку ступеня схожості інтересів користувачів та надання рекомендацій, виходячи з близькості цих інтересів. Матрицю формують наступним чином: рядки матриці відповідають користувачам інформаційної системи; стовпчики матриці відповідають послугам-об’єктам інформаційної системи; відповідна комірка перетину рядків та стовпців відображає відгук користувача інформаційної системи про дану послугу у заданому форматі(наприклад, оцінка за чисельною шкалою від 1 до 5).

Наступним етапом розробки user-user алгоритму колаборативної фільтрації, після формування означеної матриці, є етап вибору міри схожості, за якою буде оцінюватись близькість користувачів. Найпоширенішими мірами схожості, застосовуваними у рекомендаційних системах є:

- Косинусна міра;
- Коефіцієнт кореляції Пірсона;
- Коефіцієнт Тахімото;
- Евклідова відстань, тощо.

Вибір міри схожості залежить від формату даних, у якому відображається оцінка, однак не змінює подальший алгоритм роботи рекомендаційної системи.

За допомогою обраної міри проводиться оцінка схожості користувачів, після чого обраховується вихідна оцінка. Формально, обрахунок вихідної оцінки можна записати наступним чином:

$$r_{u,i} = k \sum_{u' \in U} sim(u, u') r_{u',i}$$

Де $r_{u,i}$ – прогнозована оцінка, $r_{u',i}$ – вже відома оцінка схожого користувача, $sim(u,u')$ - ступінь схожості користувачів.

Тобто, прогнозована оцінка користувача складається з добутку нормалізаційного коефіцієнта на суму добутків оцінок кожного користувача та ступеню подібності цього користувача до прогнозованого. Нормалізаціний коефіцієнт у свою чергу можна обрахувати як:

$$k = \frac{1}{\sum_{u' \in U} |sim(u,u')|}$$

Інакше кажучи, нормалізаційний коефіцієнт – величина, обернена до суми модулів мір подібності.

З наведених формул випливає, що прогнозована оцінка більшою мірою залежить від тих елементів, міра схожості з якими більша, і менше залежить від тих, міра схожості з якими менша. Виходячи з цього судження, нерідко для прогнозування вихідної оцінки застосовують спрощений алгоритм – алгоритм k найближчих сусідів.

Ідея алгоритму k найближчих сусідів полягає у обмеженні кількості елементів, що впливають на підрахунок вихідного значення до k. Елементи(у випадку user-user алгоритму – користувачі) впорядковують за спаданням міри схожості до елементу, для якого застосовують алгоритм, після чого обирають k, для яких схожість найвища.

Алгоритм k найближчих сусідів дозволяє зменшити кількість виконуваних підрахунків, зберігаючи вихідні результати близькими до бажаних.

Схему роботи user-user алгоритму продемонстровано на рисунку 1.

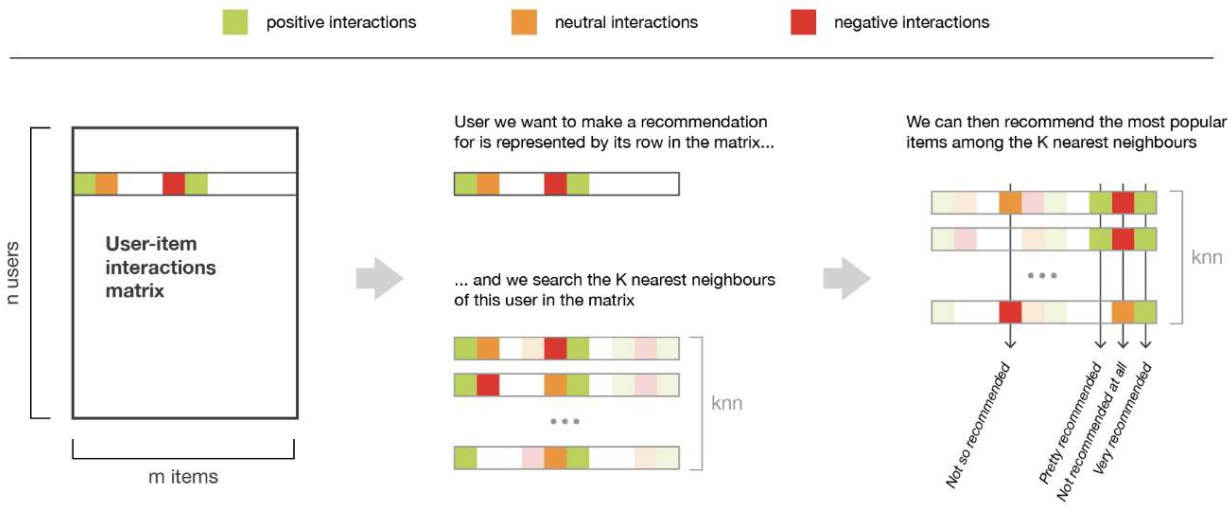

Рис. 1. Ілюстрація USER-USER алгоритму

ITEM-ITEM алгоритми

Алгоритми надання рекомендацій, виходячи з подібності послуг(item-item алгоритми), схожі до user-user алгоритмів за тим винятком, що до порівняння беруть не користувачів, а послуг.

Спершу обирають послугу, найкраще оцінену користувачем. Далі шукають для неї подібні, порівнюючи за даними з user-item матриці відповідні стовпці. У результаті роботи алгоритм повертає впорядкований за ступенем схожості перелік послуг, або k послуг, найближчих до тої, що обула найкраще оцінена користувачем.

Для підвищення якості рекомендацій можна надавати рекомендації, виходячи з кількох найкраще оцінених користувачем послуг.

Схема роботи item-item зображена на рисунку 2.

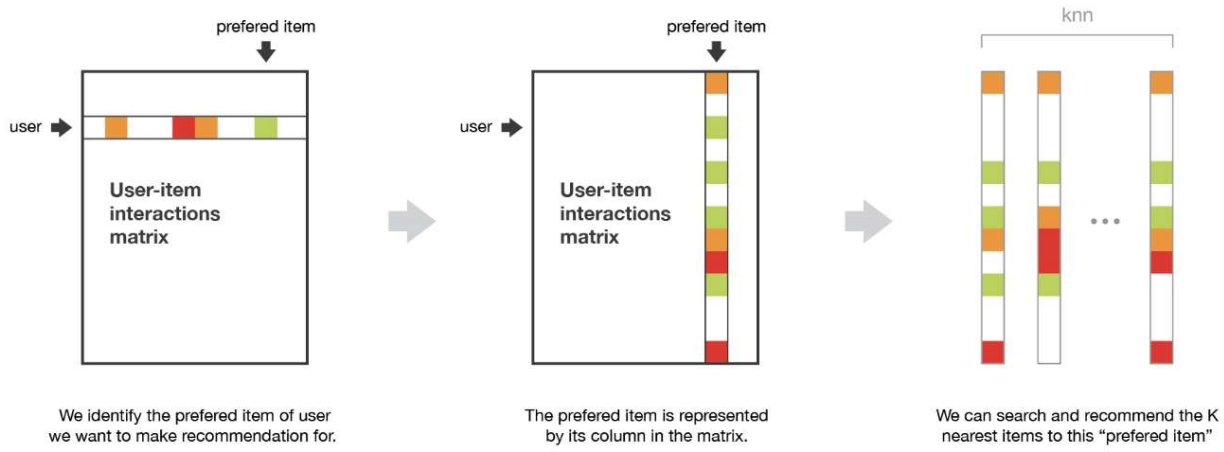

Рис. 2 Ілюстрація роботи ITEM-ITEM алгоритму

Не зважаючи на принципову схожість методів, застосовуваних у колаборативних системах user-user та item-item типу, ці методи прогнозують дещо відмінні результати. Завдяки тому, що один порівнює користувачів з користувачами, а інший послуги з послугами, вплив змін на них відображається по-різному. Кількість оцінок користувачами послуг зазвичай невисока, і кожне нове порівняння може суттєво вплинути на розподіл рекомендацій при застосуванні порівняння користувачів з користувачами. Натомість, кількість оцінок послуг користувачами зазвичай значно більша, тому нові оцінки менше впливають на вихідний результат. Тому метод порівняння користувачів з користувачами більш персоніфікований, і менше бере до уваги зв'язок послуг між собою, в той час як метод порівняння послуг з послугами надає відносно менший ступінь персоніфікації, і більший ступінь узагальненості рекомендацій. Однак сильною рисою item-item алгоритму є його менша залежність від даних, тому його результати роботи можуть бути незмінні відносно довгий період часу, а отже, можуть бути обраховані наперед, в той час, як user-user метод потребує активного перерахунку даних.

Описані методи добре працюють на рекомендаційних системах невеликих масштабів, проте зі збільшенням розміру даних час, що потребують такі системи, для проведення обрахунків, зростає пропорційно до даних. Одним з популярних методів рішення проблеми масштабування є використання методу наближених

найближчих сусідів(approximate nearest neighbors), замість методу k найближчих сусідів. Одним з варіантів реалізації такого методу може бути, наприклад, порівняння елементів лише в межах кластеру. Методи кластеризації буде розглянуто пізніше.

Іншою значною проблемою алгоритмів такої категорії може бути випадок, коли користувачі отримуватимуть рекомендації лише певної категорії товарів, найближчих до вже оцінених користувачем позитивно, що, однак, не дозволить йому дізнатись можливо не менш цікавих для нього послуг з інших категорій. Така проблема характерна для більшості рекомендаційних систем, однак для цих алгоритмів особливо, що повинно бути враховано розробником.

Matrix factorization

Як вже було зазначено вище, цей метод являє собою один з найпопулярніших методів колаборативної фільтрації. Його ідея полягає у розбитті user-item матриці на добуток матриць меншої розмірності. Метод здобув широке визнання після перемоги цього методу у конкурсі рекомендаційних алгоритмів Netflix prize.

В основу методу покладено два припущення:

1. Існують фактори, які унікально описують послуги, надавані інформаційною системою;
2. Ці ж фактори можуть бути застосовані для опису інтересів користувачів.

Однак, ми не хочемо задавати ці фактори для нашої системи і дозволяємо їй самостійно їх визначити. У процесі роботи така система може знаходити взаємозв’язки, для яких складно, а, часом, і неможливо визначити інтуїтивне пояснення, однак такі, що виражені математично. Але і зрозумілі для користувача критерії також можливі.

Зрештою, результат факторизації може бути представлений у вигляді розкладу розрідженої user-item матриці на матриці user-factor та item-factor

меншої розмірності, що являють собою матриці представлення користувачів та представлення послуг відповідно, у яких користувачі, близькі за категоріями, та схожі послуги розташовані ближче один до одного. При цьому розмірність вихідних матриць дозволяє регулювати ступінь персоналізації. Так, при розкладі матриці на вектор-стовпець та вектор, значення, яких набувають зв'язки, відповідають найбільш популярним елементам. Ідея алгоритму відображена на рисунку 3.

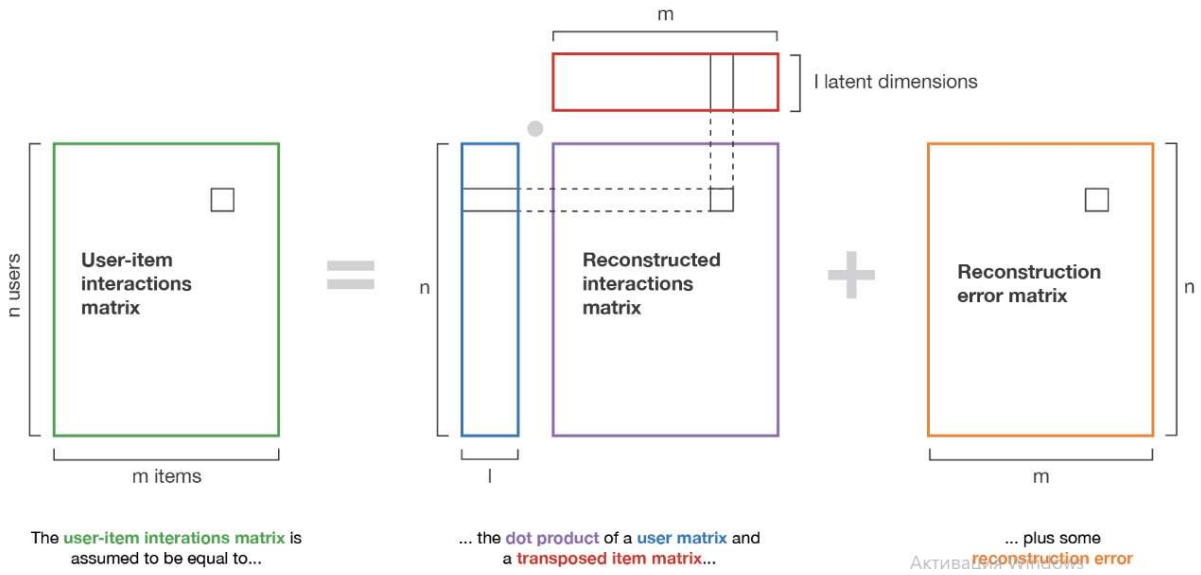

Рис. 3. Ілюстрація matrix factorization

З формальної точки зору, алгоритм базується на допущенні, будь яку матрицю можливо розкласти на добуток двох інших за формулою:

$$M = X \times Y^T$$

Тоді прогнозовану рекомендацію користувача u для послуги i можливо швидко визначити як:

$$\widetilde{r_{ui}} = \sum_{f=0}^{l} (X_{u,f})(Y_{f,i})^T$$

Відповідно, розклад шукають таким, щоб мінімізувати сумарне відхилення вихідної та отримуваної матриці:

$$(X, Y) = argmin_{X,Y} \sum [(X_u)(Y_i)^T - M_{ui}]^2$$

Однак, занадто висока розмірність призводить до перенавчання рекомендаційної системи, тому додають фактор регуляризації:

$$(X, Y) = argmin_{X,Y} \sum [(X_u)(Y_i)^T - M_{ui}]^2 + \alpha \sum_{u,f} (X_{uf})^2 + \beta \sum_{u,f} (Y_{if})^2$$

Такий обрахунок можна проводити методом градієнтного спуску, що дозволяє швидко та зручно виконувати операцію факторизації. Після застосування матричної факторизації, отримують матриці меншої розмірності, для яких, однак, можливо так само застосовувати user-user та item-item алгоритми.

Наведений алгоритм покладено в основу роботи рекомендаційної системи переможця конкурсу Netflix prize. Існують також модифікації цього алгоритму для інших систем.

Кластеризація

Пошук подібностей у багатьох методах проводять за допомогою лінійних мір, тому збільшення об’ємів порівнянь може значно зашкодити швидкодії системи. Одним зі шляхів вирішення проблеми кількості порівнянь є методи кластеризації, такі як k-means чи LSH.

Означені методи дозволяють розбити множину об’єктів на окремі підмножини, елементи яких розташовані відносно близько один до одного. Після такого розбиття, запит до системи достатньо проводити лише в межах відповідно кластеру, адже подібність елементів кластеру найвища, а отже рекомендації будуть відносно оптимальними. Для виконання цієї задачі LSH, наприклад, вводить спеціальні хеш функції, колізії яких максимізовано. Однак ми зосередимо увагу у цій роботі на алгоритмі k-means, як більш широко вживаному.

K-means

Метод к середніх дозволяє розбити дані на кластери, щоб зменшити кількість порівнянь. Об’єкти групуються за своїми характеристиками, після чого пошук можна проводити лише серед об’єктів одного кластеру.

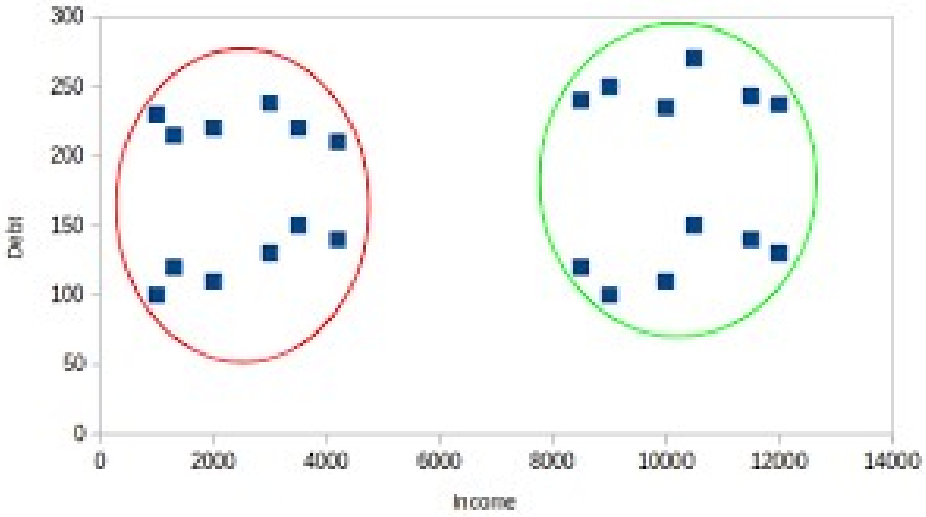

Рис. 4. Ілюстрація кластеризації

Алгоритм роботи методу наступний:

1. Довільним, або будь-яким, заснованим на даних, чином обирають k точок – центроїдів майбутніх кластерів;
2. Проводять розбиття простору даних, після чого кожному об’єкту присвоюють кластер таким чином, щоб відстань від цього об’єкту до відповідного центроїду була меншою, ніж відстань до інших центроїдів;
3. Базуючись на отриманих елементах кластерів, проводять обрахування нових центроїдів кластерів, як середнього значення всіх елементів, що належать кластерові;
4. Кроки 2 та 3 повторюють доти, поки система не стабілізується.

Алгоритм має суттєві зауваження, на які необхідно звернути увагу:

- Вибір k покладено на користувача. Знання прогнозованої кількості категорій має бути наявне ще до початку роботи алгоритму. На практиці це питання вирішують обранням довільного, достатньо

великого значення k, або впровадженням методів машинного навчання;

- У процесі роботи алгоритм досягає не глобальних, а локальних мінімумів;
- На результат роботи суттєво впливає розподіл початкових цетроїдів, однак засобами алгоритму не надається жодних рекомендацій, щодо їх вибору.

Широкої популярності також набула реалізація методу, що спирається на нейронні мережі Кохонена, яка вирішує питання вибору величини k.

Не зважаючи на наведені особливості методу, він здобув широке визнання на практиці.

Інші методи

Існує велике різноманіття рекомендаційних, окрім наведених у цій роботі. Серед таких можна навести алгоритми, засновані на баєсових мережах, логістичних методах, методах лінійної регресії, машинному навчанні. Розвиток рекомендаційних систем все ще триває, системи у реальному житті стикаються з новими вимогами надвисоких навантажень та якіснішої персоналізації, у тому числі, з урахуванням відомостей про користувача, і тому потребують значних робіт у цьому напрямку.

Висновки теоретичної частини

У розділі розглянуто відомі реалізації алгоритмів рекомендаційних систем. Серед сучасних систем особливе місце займають алгоритми колаборативної фільтрації, тому їх основним різновидам та їх підґрунтю приділено значну частину розділу. Розглянуто ідеї, закладені у основі user-user, item-item та matrix factorization алгоритмів колаборативної фільтрації. Було розглянуто і пояснено причини впровадження алгоритмів кластеризації у роботу рекомендаційних систем, з’ясовано порядок роботи широко відомого алгоритму кластеризації k середніх.

Було з’ясовано, що кожен алгоритм містить свої сильні та слабкі сторони, тому прикладні реалізації часто намагаються компенсувати недоліки, впроваджуючи комбіновані методи.

Розвиток рекомендаційних систем все ще триває, впровадження методів машинного навчання є трендовим напрямком розробки майбутніх рекомендаційних систем, викликом для яких є підвищення швидкості обробки запитів та якості персоналізації інформації, при загальному зростанні кількості даних, якими оперує система.

Практична частина

Обґрунтування обраних рішень

У роботі реалізовано кілька найпопулярніших алгоритмів рекомендаційних систем: user-user алгоритм колаборативної фільтрації, k-means алгоритм кластеризації та алгоритм рекомендацій, виходячи з популярності послуг, для покриття проблеми холодного старту.

Для виконання роботи використано дані формату конкурсу Netflix prize, як найбільш відомого конкурсу колаборативних систем.

Роботу виконано засобами мови програмування Python, як такої, що має широку можливість інтеграції у різних робочих системах та підтримує роботу з багатьма інструментами. Для збереження і роботи з даними, застосовано sql систему управління базами даних SQLite. Відповідно, основні запити до бази реалізовано мовою SQL.

Зчитування даних

Дані формату конкурсу Netflix prize зчитуються та зберігаються у СКБД SQLite. Дані представлені у наступному форматі:

- Файл movies_test.txt містить відомості про фільми, використовувані у системі. Кожен рядок файлу містить дані у форматі csv: “MovieID, YearOfRelease,Title”, де MovieID – внутрішній іденифікатор фільму у системі, YearOfRelease – рік випуску цього фільму, Title – назва фільму.
- Файл test_data.txt містить відомості про відгуки користувачів системи щодо відповідних фільмів. Файл складається з рядків двох типів. Рядки першого типу містять “MovieID:” – ідентифікатор фільму, якого стосуються відгуки, та символ двокрапки. Рядки другого типу містить дані у форматі csv: “CustomerID,Rating,Date”, де CustomerID – унікальний ідентифікатор користувача у системі,

Rating – оцінка, надана користувачем у числовому форматі від 1 до 5, Date – дата надання оцінки користувачем.

В процесі роботи скрипт створює та наповнює базу даних з таблицями, перша зберігає дані про фільми системи, а друга оцінки.

```
import sqlite3
import pathlib

print("initializing database")

MOVIE_FILE = "C:/Users/User2/Desktop/netflix-prize-
data/movies_test.txt"
DATA_FILES = ("C:/Users/User2/Desktop/netflix-prize-
data/test_data.txt",)

DATABASE_FILE = "testdata.db"

if pathlib.Path(DATABASE_FILE).exists():
    pathlib.Path(DATABASE_FILE).unlink()

conn = sqlite3.connect(DATABASE_FILE)
c = conn.cursor()

c.execute("CREATE TABLE films (movieID INT, year INT, title
TEXT)")

c.execute("CREATE TABLE rates (movieID INT, customerID INT,
rate REAL, date TEXT)")
…
```

Тестові дані для демонстрації роботи алгоритму, наведені у таблиці 1.

Таблиця 1. Вхідні тестові дані

User\Movie	1	2	3	4	5	6	7	8	9
1	5	3			4				
2	4					1		2	3
3		5	5						
4			4	3		2	1		

Прогноз за популярністю

Для вирішення проблеми холодного старту користувачів, тобто ситуації, коли дані про нового користувача відсутні, було вирішено обрати впорядкування за популярністю. Популярність розглядається як середнє арифметичне всіх відгуків, надаваних користувачами системи.

```
c.execute("select movieID, avg(rate) as popularity from rates
group by movieID order by popularity desc;")
```

Результат роботи алгоритму наведено у таблиці 2.

Таблиця 2. Впорядкування за популярністю

movie	1	3	2	5	4	9	8	6	7
popularity	4,5	4,5	4	4	3	3	2	1,5	1

User-user фільтрація

Реалізовано алгоритм колаборативної фільтрації вигляду користувач-користувач. Алгоритм використовує косинусну міру подібності для знаходження схожості користувачів та прогнозу на основі цієї подібності.

У процесі роботи скрипт створює view SIMi, де i – ідентифікатор користувача системи, для якого здійснюється прогноз. Це представлення містить дані у форматі “customerID,similarity” – відповідні пари користувача, та його подібності до порівнюваного.

Після чого всі елементи, які не представлені у відгуках поточного користувача, поєдують у представленні UNIQi, на основі даних з якого створюють остаточне представлення рекомендацій RECSi.

Робота скрипту завершується виведенням переліку рекомендацій для кожного користувача.

Лістинг цього коду наведено нижче.

```
import sqlite3
```

```
DATABASE_FILE = "testdata.db"

PROGNOSIS_COMMANDS = ("""
    drop view if exists A;
""", """
    drop view if exists B;
""","""
    drop view if exists SIM{};
""","""
    create view A as
    select * from rates where rates.customerID = {};
""","""
    create view B as
    select * from rates where rates.customerID != {};
""","""
    create view SIM{} as
    select UP.customerID as customerID,
sum(UP.rate/BOT1.rate/BOT2.rate) as similarity from
     (select sum(A.rate * B.rate) as rate, B.customerID as
customerID from
       A, B
       where A.movieID = B.movieID
       group by B.customerID
     ) as UP,
     (select sum(A.rate * A.rate) as rate from A)
     as BOT1,
     (select sum(B.rate * B.rate) as rate, B.customerID as
customerID from B
     group by B.customerID
     ) as BOT2
    group by UP.customerID order by similarity desc;
""","""
    drop view if exists UNIQ{}
""",
"""
    create view UNIQ{} as
    select * from rates
     where
      rates.customerID in
      (select customerID from SIM{})
     and
      rates.movieID not in
      (select movieID from rates where rates.customerID = {})
""","""
```

```
    drop view if exists RECS{}
""","""
    create view RECS{} as
    select U.movieID as movieID, U.rate*S.similarity as rec
    from UNIQ{} as U, SIM{} as S
    where U.customerID = S.customerID
""","""
    select R.movieID, R.rec/(sum(S.similarity))
    from RECS{} as R, SIM{} as S
    group by R.movieID
""")

def prognosis(custID):
    return list(map(lambda x: x.format(custID, custID, custID)
if "{}" in x else x, PROGNOSIS_COMMANDS))

conn = sqlite3.connect(DATABASE_FILE)
c = conn.cursor()

c.execute("select min(customerID) as minID, max(customerID) as
maxID from rates;")
minID, maxID = list(map(int, c.fetchone()))

for i in range(minID,maxID+1):

    with conn:
        for comm in prognosis(i):
            #print (comm)
            c.execute(comm)

    print ("customer ", i," recs:\t",c.fetchall())

c.close()
conn.close()
```

Результат роботи алгоритму наведено у таблицях 3-6.

Таблиця 3. Прогноз вподобань для користувача 1

movie	3	9	8	6
prognosis	2,143	1,714	1,143	0,571

Таблиця 4. Прогноз вподобань користувача 2

movie	5	2	3	4	7
prognosis	3,636	2,727	0,364	0,273	0,091

Таблиця 5. Прогноз вподобань користувача 3

movie	1	4	5	6	7
prognosis	2,143	1,174	1,174	1,143	0,571

Таблиця 6. Прогноз вподобань користувача 4

movie	2	1	9	8
prognosis	4,545	0,364	0,272	0,182

Як бачимо, хоча нам і вдалось отримати персоналізований прогноз, однак, найімовірніше, більшість користувачів будуть не задоволені такими результатами прогнозу. В той час, як користувач отримав достатньо позитивний прогноз, розрідженість даних спричинила низьку впевненість прогнозування для інших користувачів.

K-means

Скрипт реалізує алгоритм, описаний у основній частині курсової роботи, у розділі алгоритмів кластеризації. Загальна ідея алгоримту полягає у розбитті користувачів на кластери довільним чином, та поступовий перерахунок центрів цих кластерів, доки не буде досягнуто точки рівноваги.

```
import sqlite3

DATABASE_FILE = "testdata.db"
K = 2
def cyclon_generator(k):
    while True:
        for i in range(0, k):
            yield i

cyclon = cyclon_generator(K)

def next_cyclon():
    return next(cyclon)

conn = sqlite3.connect(DATABASE_FILE)
```

```
conn.create_function("cyclon", 0, next_cyclon)

c = conn.cursor()

c.executescript("""
drop table if exists INIT_CL;
create table INIT_CL as
    select R.movieID as mID, R.customerID as cID, R.rate,
CLAST.claster from rates as R,
     (select customerID, cyclon() as claster from rates group
by customerID) as CLAST
    where CLAST.customerID = R.customerID;
""")
while True:

    c.executescript("""

    drop view if exists CLASTER_MEANS;
    create view CLASTER_MEANS as
    select claster, mID, cID, avg(rate) as rate from INIT_CL
group by mID;
    """)
```

Скрипт проінінціював таблицю INIT_CL, присвоївши користувачам у порядку появи користувачів у rates матриці відповідні кластери.

Представлення CLASTER_MEANS відповідає центроїдам обрахованих кластерів.

```
    SIMILARITY_COMMANDS = ("""
        drop view if exists A;
    """, """
        drop view if exists B;
    ""","""
        create view A as
        select claster, mID as movieID, cID as customerID,
rate from CLASTER_MEANS where CLASTER_MEANS.claster = {};
    ""","""
        create view B as
        select * from rates;
    ""","""
        drop view if exists CONTROL_STAGE{};
```

```
    ""","""
        create view CONTROL_STAGE{} as
        select UP.customerID as customerID,
sum(UP.rate/BOT1.rate/BOT2.rate) as similarity, BOT1.claster
from
         (select sum(A.rate * B.rate) as rate, B.customerID as
customerID from
           A, B
           where A.movieID = B.movieID
           group by B.customerID
         ) as UP,
         (select sum(A.rate * A.rate) as rate, claster from A)
         as BOT1,
         (select sum(B.rate * B.rate) as rate, B.customerID as
customerID from B
          group by B.customerID
          ) as BOT2
        group by UP.customerID order by similarity desc;
    """)

    def similarity(custID):
        return list(map(lambda x: x.format(custID) if "{}" in
x else x, SIMILARITY_COMMANDS))
```

SIMILARITY_COMMAND – група команд, що підраховують близькість користувачів до центроїдів відповідних кластерів.

```
    c.execute("select min(claster) as minID, max(claster) as
maxID from CLASTER_MEANS;")
    minID, maxID = list(map(int, c.fetchone()))

    c.execute("drop table if exists MATCHER")
    c.execute("create table MATCHER (customerID, similarity,
claster)")


    for i in range(minID,maxID+1):

        with conn:
            for comm in similarity(i):
                c.execute(comm)

        c.execute("insert into MATCHER select * from
CONTROL_STAGE"+str(i))
```

Таблиця MATCHER об’єднує всі дані про відстань від користувачів до центроїдів кластерів.

```
    c.execute("drop view if exists REPLACER;")
    c.execute("""
        create view REPLACER as
        select PAT.customerID, MATCHER.claster from
        (select customerID, max(similarity) as similarity from
MATCHER group by customerID) as PAT, MATCHER
        where PAT.customerID=MATCHER.customerID and
PAT.similarity = MATCHER.similarity
         """)
```

Після чого представленню REPLACER передають інформацію щодо найближчих центроїдів у вигляді пар (користувач, центроїд).

```
c.execute("select count(*) from INIT_CL as I inner join
REPLACER as R on I.cID = R.customerID where I.claster !=
R.claster")

    amo, = c.fetchone()
```

Здійснюють перевірку, чи мають змінити користувачі кластери.

```
    if (amo > 0):
        c.execute("""
        update INIT_CL
        set claster= (select claster
        from REPLACER
        where INIT_CL.cID = REPLACER.customerID)
        """)
        print("recount(",amo,")")
        continue;
    break;

c.close()
conn.close()

print ("clusterization done")
```

Якщо мають, змінити початковий розподіл та провести перерахунок. Якщо ні – закінчити етап кластеризації. Тепер пошук рекомендацій можна здійснювати в рамках лише одного кластеру, до якого належить користувач, адже користувачів об'єднано з міркувань близькості:

```
c.execute("""
select mID, avg(rate) as popularity
from INIT_CL
where INIT_CL.claster = (select claster from INIT_CL where cID
= {})
group by mID
order by popularity desc
    """.format(USER))
```

Результат роботи алгоритму кластеризації наведено у таблиці 7.

Таблиця 7. Результат кластеризації

User\Movie	1	2	3	4	5	6	7	8	9	cluster
1	5	3			4					1
2	4					1		2	3	2
3		5	5							3
4			4	3		2	1			3
5	5	3			3					1
6	4	2								1
7			5						3	3
8		5		3						1

Висновки практичної частини

У розділі розглянуто реалізацію кількох найпоширеніших алгоритмів та їх роботу на тестових даних. У процесі тестування на практиці було підтверджено вразливі місця, що виникають при експлуатації цих алгоритмів. Їх наявність підштовхує до подальшого пошуку рішень комбінативного характеру, для компенсації сильними сторонами одних алгоритмів, слабких сторін інших.

Висновки

У межах цієї курсової роботи було розглянуто та реалізовано деякі з популярних методів рекомендаційних систем. Особливу увагу було приділено алгоритмам колаборативної фільтрації та k-means як таких, що мають найбільше коло застосування. У процесі їх тестування було на практиці підтверджено необхідність впровадження комбінованих підходів до вирішення задачі надання рекомендацій, для компенсації слабких сторін відповідних рекомендаційних алгоритмів.

Огляд сучасного стану рекомендаційних систем дав зрозуміти, що, не зважаючи на вже значну кількість зусиль, спрямованих у напрямку їх розробки у світі, поки не існує універсальних рішень поставлених перед ними задач, тому можна ще очікувати розвиток цього напряму і впровадження більш новітніх підходів.

Використані джерела

1. Стаття Introduction to recommender systems https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
2. Стаття How does Netflix or Spotify knows what you like? – A briefing on Recommender Systems https://leantechblog.wordpress.com/2018/01/03/how-does-netflix-or-spotify-knows-what-you-like-a-briefing-on-recommender-systems/
3. Стаття A Taxonomy of Recommender Agents on the Internet
4. https://www.researchgate.net/publication/220637828_A_Taxonomy_of_Recommender_Agents_on_the_Internet
5. Лекція Introduction to Recommender Systems
6. http://technocalifornia.blogspot.com/2014/08/introduction-to-recommender-systems-4.html
7. Стаття Do We Really Need Machine Learning for Personalized Recommendations? https://celadonsoft.com/ai-ml/machine-learning-for-personalized-recommendations
8. Стаття Recommender systems explained https://medium.com/recombee-blog/recommender-systems-explained-d98e8221f468#.mgj1g23fn
9. Data Mining Methods for Recommender Systems https://www.researchgate.net/publication/225924875_Data_Mining_Methods_for_Recommender_Systems