УДК 004.043.93’12

Пєчкурова О. М.

ЛОКАЛЬНО-ЧУТЛИВЕ ХЕШУВАННЯ ТА СФЕРИ ЙОГО ЗАСТОСУВАННЯ

На сьогодні існує безліч технологій для роботи з даними невеликих розмірів, вони виконують поставлені їм завдання швидко та якісно. Проте вони є неефективними для великих масивів даних, особливо якщо мова йде про знаходження подібностей. Для порівняння мільярдів чи навіть трильйонів множин потрібен метод чи технологія, яка б робила акцент на тих парах множин, що можуть бути дуже схожими між собою, при цьому ігноруючи переважну більшість інших пар. У статті описано особливості використання локально-чутливого хешування, яке здатне вказувати на подібні пари, не пробираючись через безліч усіх інших пар.

Ключові слова: локально-чутливе хешування, хеш-функція, шингл.

Розглянемо конкретні важливі застосування знаходження лексично подібних документів у великій колекції документів чи файлів. Матимемо на увазі не файли з однаковими темами, а подібну послідовність символів між ними. Подібні технології використовуються, наприклад, для знаходження дзеркал веб-сторінок, плагіату. Ще одне застосування стосується сайтів, що публікують нові історії. Стаття може бути репродукована одним джерелом і поширена тисячею сайтів з новинами, кожен з яких робитиме певні модифікації цієї статті. Для агрегатора важливо розуміти, що веб-сторінки говорять про ту саму історію, оскільки вони походять з одного й того самого оригіналу, навіть якщо вони були якимось чином змінені.

Основні технології для знаходження «подібних» документів – це шинглінг – конвертування документів у множини, мінхешинг – конвертування множин у короткі сигнатури (можемо поглянути на сигнатури двох множин і приблизно сказати, наскільки схожими є множини, отримані після шинглінг-процесу), локально-чутливе хешування дає нам змогу уникнути перегляду всіх пар сигнатур, які не є схожими наборами.

Локально-чутливе хешування (locality-sensitive hashing)

Головна ідея цього підходу полягає у створенні колекції всіх елементів (у нашому випадку сигнатур), чиї подібні пари ми хочемо знайти, не створюючи при цьому всі можливі пари цих елементів із невеликого списку пар-кандидатів. При створенні пар-кандидатів ми повинні дивитися тільки на індивідуальні елементи, а не на самі пари. Пари, що не є парами-кандидатами, вважаються не схожими, хоч в окремих випадках це твердження може бути помилковим. Такі пари можуть бути схожими, проте вони ніколи не будуть проходити перевірку на подібність [1].

Для матриць сигнатур ми виконуємо локально-чутливе хешування створенням великої кількості хеш-функцій (звичайних хеш-функцій, не мінхеш-функції). Для кожної вибраної хеш-функції ми хешуємо стовпці в кошики. Для кожного кошика шукаємо пари-кандидати. Пара стає кандидатом, якщо хоча б одна або більше хеш-функцій кладуть обидві сигнатури в один і той самий кошик. Ми повинні налаштувати кількість хеш-функцій і кількість кошиків для кожної хеш-функції так, щоб у кошиках була відносно невелика кількість сигнатур. Отже, генерується невелика кількість пар-кандидатів. Проте ми не можемо використовувати занадто багато кошиків, оскільки пари, які є дійсно схожими між собою, не зможуть потрапити в один і той самий кошик хоча б після однієї хеш-функції, що ми використовуємо [3].

Розглянемо приклади того, як локально-чутливе хешування використовують на практиці. Для кожного окремого прикладу ми будемо змушені змінювати та використовувати технології так, щоб вони допомагали нам вирішити індивідуальні проблеми, які виникатимуть. Тож трьома прикладами, які ми будемо розбирати, є:

1. Розпізнавання сутностей – досліджує та перевіряє на збіг дані записів, які відносяться до однієї сутності.

2. Відповідність відбитків пальців. Відбитки пальців можна представити у вигляді множин (наборів). Проте нам потрібно дослідити інший набір локально-чутливих хеш-функцій, що можна використовувати замість функції, яка отримується за допомогою технології мінхешингу.

3. Відповідність газетних статей. Тут акцентується увага на основних статтях веб-сторінки інтернет-газети, ігноруючи будь-які інші сторонні матеріали.

Розпізнавання сутностей

Розглянемо колекцію записів. Кожен запис містить інформацію про певний конкретний об'єкт (зазвичай такими об'єктами є люди, проте це можуть бути і компанії, фізичні місця, події чи будь-що інше). Проблема розпізнавання сутностей полягає у визначенні набору записів, що відносяться до конкретної особи, та в злитті всіх таких записів в один запис, який міститиме всю інформацію про цю сутність. Ця проблема є набагато важчою, ніж здається на перший погляд. Наприклад, зрозуміло, що записи про людей містять імена. Отже, видається, що згрупувати їх у набори, які представляють одну особу, взагалі не має становити жодних труднощів. Проте у великій колекції записів можуть бути люди з однаковими іменами, тому групування за іменем може злити записи про багатьох різних людей. Можуть бути ситуації, коли ім'я однієї людини може бути написано по-різному в декількох записах, або «нікнейм» людини може з'являтися в одних записах, а офіційне ім'я в інших (наприклад, Соня і Софія). І, звісно ж, можуть з'являтися орфографічні помилки, які роблять імена на вигляд різними, незважаючи на те, що насправді вони відносяться до однієї і тієї самої особи.

Часто можемо компенсувати такі розбіжності, використовуючи іншу інформацію із записів. Наприклад, два записи можуть мати схожі імена (назви), але ідентичні номери телефонів чи адреси. У такому разі проблема розпізнавання сутностей стає дійсно цікавою.

Уявімо, у нас є дві компанії A і B, між якими виникла така суперечка: компанія B була сервіс-центром, для якої компанія A, використовуючи власну клієнтську базу, шукала клієнтів. У певний момент угоду між двома компаніями було розірвано. Після цього компанія B повинна була платити компанії A за кожного клієнта, знайденого компанією A. Проте цього не відбувалося, і компанія A подала в суд. На жаль, жодна з компаній не вказувала у власних записах, який клієнт частиною якої угоди був і клієнтом якої компанії він був із самого початку. Тож проблема полягає в порівнянні записів із двох наборів для розуміння того, чи пара представляє одну людину чи ні. Для розуміння масштабу проблеми потрібно згадати про той факт, що кожна компанія має приблизно мільйон записів, які можуть представляти клієнта, якого компанія A знайшла для компанії B. За сьогоднішніми стандартами це крихітна база даних, проте ми маємо розуміти, що існує трильйон пар записів із компанії A та з компанії B, що можуть бути однією сутністю. Кожен запис із будь-якої компанії складається з імені, адреси і номера телефону, проте часто вони можуть бути різними навіть для однієї людини (через безліч можливих факторів).

Таким чином, перші наші дії мають бути такими: ми повинні придумати міру схожості записів. Надамо 100 балів кожному за однакові імена, адреси та номери. Отже, 300 буде найвищим балом. Цікаво, що тільки 7000 пар записів одержать таку оцінку, хоч і існує більше ніж 180 тисяч пар, які мали дуже високу ймовірність бути однією й тою самою людиною. Далі ми штрафуємо за відмінності в цих трьох полях. Абсолютно різні імена, адреси чи телефони отримують 0. Проте, якщо є якісь незначні відмінності в будь-якому з цих полів – то оцінка для них близька до ста. Оцінюємо так усі пари-кандидати записів і повідомляємо ті пари, які перевищують певний поріг. Тут виникає нюанс, як ми встановили поріг, не знаючи, які пари записів насправді були створені одними й тими самими особами.

Тож скористаємося трьома хеш-функціями. Перша має кошик для всіх можливих імен. Друга – для всіх можливих адрес і третя – для всіх можливих номерів. Тепер пари кандидатів будуть ті, що були розміщені в одному й тому самому кошику за допомогою хоча б однієї з цих хеш-функцій. Звичайно, деякі пари загубляться, оскільки існують такі пари записів, що мають небагато відмінностей у кожному з цих трьох полів, і вони ніколи не стануть кандидатами для підрахунку (кількість таких пар приблизно 2500).

Може виникнути спантеличення тим, що ми хешуємо в один кошик кожне можливе ім'я, хоча можливих імен може бути нескінченна кількість. Проте в цьому випадку ми швидше сортуємо записи за іменем, і потім записи з однаковими іменами послідовно з'являються в списку, де ми можемо підрахувати кожну пару з ідентичними іменами. Далі аналогічну процедуру застосовуємо до адресата номерів. Слід зауважити, що ми могли хешувати в декілька мільйонів кошиків і порівнювати всі пари записів в одному кошику. Це б деколи змушувало нас дивитися на пари записів з різними іменами, які були відхешовані до одного кошика,

проте якщо кількість кошиків є набагато більшою за кількість різних імен, що б насправді з'явилися, то ймовірність таких колізій була б дуже низькою.

Пригадаємо, що ми дали оцінку кожній парі-кандидату записів, та припустимо, що пара отримує 200 балів із 300, що свідчить про те, що схожість є чималою, але не ідеальною. Чи такі записи представляють одну й ту саму людину? Виявляється, що якщо пара отримала оцінку 200, то дуже ймовірно, що записи, які входять у цю пару, відносяться до однієї людини.

Проте, як можна бути впевненим у цьому? Ми розробили спосіб обчислення ймовірності того, що записи з оцінкою **x** представляють одну людину. Згадаємо про існування золотого стандарту. Існує 7000 пар ідентичних записів, які, за нашим припущенням, стосуються однієї людини. Для цих пар ми дивимось на дату створення в компаніях A і B. Виявляється, в середньому було 10 днів затримки між тим, як запис був створений у компанії A, та між тим, як та сама особа відправилася в компанію B і там почала користуватися їхнім сервісом. З іншого боку, щоб зменшити кількість пар записів, ми дивимось тільки на ті записи, які створила компанія A менше ніж за 90 днів до створення таких самих записів у компанії B. Тож, якщо ми візьмемо будь-який такий запис із компанії A та будь-який такий запис із компанії B, то ми отримаємо середню затримку в 45 днів. Оскільки ці записи були вибрані у випадковому порядку, вони майже точно представляють різних людей.

Давайте подивимось на набір збігів з оцінкою 200. Для деяких зіставлень, які є дійсними, 10 днів буде середньою різницею в днях створення. Для інших, які є помилковими, це число дорівнюватиме 45 дням. Припустимо, що в цьому наборі збігів середня різниця дорівнює X. Тобто частка дійсних збігів дорівнюватиме (45 – X)/35. Наприклад, якщо X = 10, то ця частка дорівнює одиниці. Це має сенс, оскільки 10 – це різниця, яку забезпечує золотий стандарт. Якщо X = 20, то ми можемо очікувати, що 5/7 збігів є дійсними.

Відповідність відбитків пальців

У цьому прикладі застосування локально-чутливого хешування існує велика колекція відбитків пальців, з якої потрібно знайти, які саме пари належать певній людині. Аналіз відбитків пальців не потребує знаходження всіх пар одночасно. Він полягає в тому, що існує база даних з усіма відомими відбитками, і коли новий відбиток з'являється, відбувається порівняння його з усіма іншими відбитками, що були в базі попередньо. У цьому випадку технологія локально-чутливого хешування є чудовим способом організувати базу даних таким чином, щоб можна було переглядати, чи є збіг з новим відбитком усього в декількох кошиках.

Для початку ми повинні знати про те, як відбитки пальців представлені. Зображення відбитка пальця перевіряється на minutiae (дрібниці). У цьому контексті minutiae – це місця, де відбувається щось незвичне, наприклад, закінчення чи об'єднання ліній. Таким чином, зображення відбитка пальця можна замінити набором координат у двовимірному просторі, де містяться «деталі». Ви розміщуєте сітку над кожним зображенням відбитка пальця. Ця сітка має бути масштабована і орієнтована належним чином для того, щоб, якщо ми маємо два зображення одного й того самого відбитка пальця різного розміру, то сітки повинні перекриватися. Далі кожен відбиток представляється множиною квадратів сітки, які містять деталі. Оскільки деякі деталі будуть прямо на межі або недалеко від межі (кордону), було б корисно розглядати такі деталі присутніми у квадратах із двох боків кордону. Отже, ми звели задачу знаходження збігів відбитків пальців до задачі знаходження схожих множин сітки квадратів, що містять деталі. Проблема полягає в нерозрідженості результуючої матриці. Сітка не може бути занадто тонкою, оскільки в такому разі буде незрозуміло, куди саме деталь належить. Тобто в цьому випадку мінхешинг працюватиме не дуже добре.

Припустимо, що деталі типового відбитка пальця займають 20 % на сітці квадратів і що, якщо два відбитки пальців представляють один палець, тоді принаймні 80 квадратів із деталями одного мають деталі з іншого відбитка. Якщо відбитки пальців є з різних пальців, то ймовірність того, що обидва відбитки містяться в одному кошику, є дуже малою. Для кожного пальця кожен відбиток має 20 % шанс мати деталі в кожному квадраті. Отже, шанс для нього мати деталі в усіх квадратах $(0.2)^3$. Ймовірність того, що кожен із двох рандомних відбитків пальців одночасно матиме деталі в усіх трьох квадратах = $(0.2)^6$ = .000064.

Наразі поглянемо на два відбитки одного пальця. Ймовірність для них двох бути в одному кошику є набагато вищою. Причина в тому, що є багато кореляцій між кошиками, які міститимуть ці відбитки. Для початку для будь-якої заданої сітки квадрата ймовірність того, що

перший відбиток має якісь деталі, становить 0.2. Ймовірність того, що другий має – 0.8. Оскільки є три квадрати, кожен з яких повинен містити деталі з обох відбитків, нам потрібно провести такі обчислення: $((0.2)(0.8))^3 = .004096$. Ця ймовірність є в 64 рази більшою від імовірності, коли відбитки походять від двох різних пальців.

Потрібно пам'ятати, що ми маємо 1024 множини з трьох квадратів. Для того щоб пара відбитків могла стати парою-кандидатом, нам лише потрібно знайти їх разом в одному з цих 1024 кошиків. Імовірність того, що це станеться хоча б один раз, дорівнює 98,5 %.

Відповідність газетних статей

Суть цієї проблеми полягає в знаходженні статей новин, що представляють однакову історію. Оскільки одну історію можуть використати багато джерел новин і кожне з них репрезентує її по-іншому, ця задача є схожою на проблеми пошуку подібних документів загалом. Проте ми не можемо використовувати технологію шинглінгу, яку ми описували вище, оскільки вона відноситься до всіх частин документа однаково. А ми хочемо мати можливість ігнорувати деякі частини документа (як-от реклама чи заголовки інших статей), на які газета чи веб-сторінка дає посилання і які не є частиною інформаційної статті.

Виявляється, існує помітна різниця між текстом, який з'являється в основній частині статті, та між текстом, який з'являється в заголовках чи в рекламі. Основна частина статті має набагато більшу частку стоп-слів, як-от «і» або «але». Загальна кількість стоп-слів змінюється залежно від застосунку, проте прийнято використовувати список, що складається з декількох сотень найбільш часто вживаних слів.

Типове рекламне оголошення може просто звучати так: «Купляйте Ariel». Повна версія цієї ж думки, яка може з'явитися в статті, може звучати так: «Я рекомендую вам купити Ariel для вашого одягу». У цьому випадку нормальною практикою буде ставитися до таких слів, як «я», «вам», «для» і «вашого», як до стоп-слів.

Припустимо, ми визначили шингл як стоп-слово з двома словами, які йдуть після нього. З цього випливає, що оголошення «Купляйте Ariel» немає жодного шингла і не буде відображене в представленні веб-сторінки, на якій міститься це оголошення. З іншого боку, речення «Я рекомендую вам купити Ariel для вашого одягу» буде представлене чотирма шинглами: «Я рекомендую вам», «вам купити Ariel», «для вашого одягу» та «вашого одягу х», де х – будь-яке слово, яке слідує в цьому реченні.

Припустимо, ми маємо дві веб-сторінки, кожна з яких складається наполовину з новин і наполовину з реклами чи іншого матеріалу, що містить невелику кількість стоп-слів. Якщо текст новин є однаковим для двох сайтів, але інший матеріал відрізняється, то можна очікувати, що більша частина шинглів двох веб-сторінок буде однаковою. Вони можуть мати «жаккардову схожість» 75 %. Утім, якщо довколишній матеріал є однаковим, але контент новин відрізняється, то кількість спільних для двох веб-сторінок шинглів буде невеликою, десь 25 %. Якщо б ми використовували звичайний метод шинглінгу, де шингли були б послідовностями з 10 символів поспіль, ми б могли очікувати жаккардову схожість 1/3, незалежно від того, чи це були б новини чи навколишній матеріал [2].

Висновки

Отже, можемо сказати, що локально-чутливе хешування є ефективним методом для роботи з даними великих розмірів, адже він за дуже короткий час може знайти схожі елементи. Метод досліджує не всю множину елементів, а переглядає тільки ті елементи, що мають імовірність бути схожими. Тобто локально-чутливе хешування зосереджується на схожих парах елементів (парах-кандидатах), не досліджуючи кожну пару. Проте, якщо нашою метою є дослідження схожості кожної пари, то локально-чутливе хешування для цього не підходить.

Список літератури

1. Andoni A. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions [Electronic resource] / Alexandr Andoni, Piotr Indyk. – Mode of access: http://web.mit.edu/andoni/www/LSH/index.html. – Title from the screen.
2. Leskovec J. Mining of Massive Datasets / Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman. – Cambridge University Press, 2014. – 513 p.
3. LSH Algorithm and Implementation (E2LSH) [Electronic resource]. – Mode of access: http://web.mit.edu/andoni/www/LSH/index.html. – Title from the screen.

O. Pyechkurova

LOCALITY-SENSITIVE HASHING AND ITS SCOPE

To date, there are many technologies for working with data of small sizes; they perform their tasks quickly and qualitatively. However, they are ineffective for large amounts of data, especially when it comes to finding similarities. To compare billions or even trillions of sets requires a method or technology that would focus on those pairs of sets that can be very similar to each other, while ignoring the vast majority of other pairs. To do this, a locality-sensitive hashing was invented, which is capable of pointing to such pairs without getting through the swamp of all other pairs.

The method does not examine the whole set of elements; rather, it considers the elements that are likely to be similar. That is, locality-sensitive hashing focuses its attention on similar pairs of elements (candidate pairs) without exploring each pair. However, if the goal is to study the similarity of each pair, then locality-sensitive hashing does not work for this.

Keywords: locality-sensitive hashing (LSH), hash function, shingle.

Матеріал надійшов 12.09.2017

УДК 004:002.1-028.27

Борозенний С. О.

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ФОРМАТУ EPUB 3 ДЛЯ СТВОРЕННЯ ЕЛЕКТРОННИХ ПУБЛІКАЦІЙ

У статті викладено підходи до створення електронних публікацій технічних текстів у форматі epub3, наведено приклади створення елементів публікацій з використанням HTML5, MathML, крім того, розглянуто проблеми, що виникають при їх створенні, та шляхи вирішення.

Ключові слова: EPUB 2, EPUB 3, MathML, електронна публікація.

Вступ

EPUB 2 (англ. electronic publication) – відкритий стандарт формату електронних книг, що відповідає стандарту International Digital Publishing Forum (IDPF) [1].

EPUB 2 став офіційним стандартом IDPF у вересні 2007 року, замінивши старий стандарт Open eBook. Може скластися враження, що EPUB 2 використовується лише для публікації художньої літератури, але насправді EPUB є загальноприйнятим форматом електронного документа і може бути використаний для представлення багатьох видів публікацій: журналів, газет і т. п., тобто будь-який тип документа, який ви хочете розповсюдити в електронному вигляді, може бути представлений як EPUB 2.

EPUB 2 забезпечує всі можливості форматування та макетування HTML 4 та CSS 2, що цілком достатньо для текстових публікацій. Проте відомо, що EPUB 2 не найкраще рішення для створення мультимедійних книг, книг зі складним макетом, математичних публікацій та інтерактивних документів.

Інструментом, що дає змогу ефективно розв'язувати такі проблеми, є новий стандарт електронних публікацій EPUB 3.