KHAI THÁC TOP-K ĐỒNG XUẤT HIỆN SỬ DỤNG BITTABLE

# Tổng quan

# Cơ sở lý thuyết

# Giới thiệu

Khai thác tập phổ biến được đề xuất đầu tiên bởi Agrawal et al. nhằm phân tích giỏ hàng để rút ra tất cả các luật liên kết. Kể từ khi việc khai thác dữ liệu này được đề xuất và những thuật toán khai thác hữu ích được liên kết của nó, đã có hàng nghìn công trình nghiên cứu theo sau trên nhiều loại mở rộng và ứng dụng. Khai thác tập phổ biến đã xuất hiện như một chủ đề quan trọng trong lĩnh vực khai thái dữ liệu. Nó đã được chứng minh đóng một vai trò thiết yếu trong nhiều nhiệm vụ khai thác dữ liệu như mining associations, correlations, causality, sequential itemsets, episodes, multi-dimensional itemsets, max-itemsets, partial periodicity and emerging itemsets

Trong khai thác tập phổ biến, một tập là phổ biến nếu tuần suất xuất hiện của nó trong cơ sở dữ liệu không nhỏ hơn một ngưỡng cho trước. Có nghĩa là, tập sự kiện phổ biến là một khái niệm bao trùm trong phạm vi tất cả cơ sổ dữ liệu mà không liên quan đến bất kỳ tập sự kiện nào khác . Tuy nhiên, trong một số ứng dụng như các hệ thống gợi ý hoặc mạng xã hội, người ta có lẻ quan tâm đến những sự kiện có liên quan với nhau nhiều hơn. Ví dụ trong hệ thống đưa ra gợi ý của một cửa hang trực tuyến, bất cứ khi nào một người dung thanh toán một vài sản phẩm, hệ thống phải đưa ra những gợi ý về những sản phẩm khác mà thường mua cùng với những sản phẩm đó để bán thêm nhiều sản phẩm. Những việc đó có thể được minh họa bằng ví dụ dưới đây.

Ví dụ: hãy xét một cơ sở dữ liệu chứa những giao dịch mua hàng được thể hiện ở bảng 1. Bảng 1 mô tả một cơ sở dữ liệu giao tác. Cơ sở dữ liệu này sẽ là đầu vào cho ví dụ cũng như mô phỏng thuật toán của toàn bộ đề tài. và Items là những thuộc tính thuộc miền sau và . Mỗi giao tác trong có một định danh duy nhất () là một tập con của . Để đơn giãn, chúng ta thể hiện một giao tác với bởi . Ví dụ là ký hiệu cho giao tác thứ hai trong Bảng 1. Top-2 những sự kiện đồng xuất hiện với tập sự kiện là sự kiện và vì có ba giao tác mua hàng chứa và hai giao tác mua hàng chứa trong khi những sự kiện khác đồng xuất hiện với hầu hết có một giao tác. Tương tự, top-2 sự kiện đồng xuất hiện với tập sự kiện là sự kiện và .

Bảng 1: Ví dụ cơ sở dữ liệu giao tác

|  |  |
| --- | --- |
| TID | Items |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

Mặc dù nhiều thuật toán đã được đề xuất để khai thác các tập sự kiện phổ biến và top-k tập sự kiện phổ biến, nhưng không khả thi để dễ dàng áp dụng những thuật toán đó để khai thác top-k sự kiện đồng xuất hiện. Những lý do có thể được liệt kê sau đây:

Khai thác top-k sự kiện đồng xuất hiện không cần thiết lập một ngưỡng hổ trợ nhỏ nhất như khai thác tập sự kiện phổ biến hay top-k tập sự kiện phổ biến, nơi mà một tập sự kiện phổ biến được định nghĩa như một tập có độ hổ trợ không nhỏ hơn ngưỡng hổ trợ nhỏ nhất cho trước. Vì thế, không thể áp dụng trực tiếp các thuật toán cho khai thác tập sự kiện phổ biến hay top-k tập sự kiện phổ biến vào khai thác top-k sự kiện đồng xuất hiện khi mà không khả thi để set một ngưỡng phù hợp để tìm top-k sự kiện đồng xuất hiện của mỗi tập sự kiện. Chú ý rằng, độ hổ trợ tối thiểu của top-k sự kiện đồng xuất hiện của những tập sự kiện khác nhau thì vô cùng, chúng ta không thể nào biết trước được.

Cho một tập sự kiện truy vấn, một hướng tiếp cận đơn giản để tìm top-k những sự kiện đồng xuất hiện là xem xét tất cả những sự kiện không liên quan đến những tập sự kiện trong cơ sở dữ liệu. Chắc chắn cách tiếp cận này sẽ gặp phải vấn đề không gian tiềm kiếm lớn, đặc biệt khi cơ sở dữ liệu rất lớn và một con số lớn của những tập sự kiện truy vấn cần được tìm kiếm top-k những sự kiện đồng xuất hiện của chúng. Vì thế, để giảm không gian tiềm kiếm và tìm top-k những sự kiện đồng xuất hiện một cách hiệu quả là một thách thức mới.

Cho là tập tất cả các sự kiện và là một cơ sở dữ liệu giao tác, với mỗi là một giao tác là tập các sự kiện mà . Chúng ta cũng có thể gọi là một itemset nếu là tập các sự kiện .

**Định nghĩa 1:** Cho là một itemset. Một giao tác được gọi là chứa nếu và chỉ nếu .

**Định nghĩa 2:** Độ hổ trợ của một itemset , được ký hiệu là , là số giao tác chứa trong .

**Định nghĩa 3:** Cho itemset , một sự kiện là một sự kiện đồng xuất hiện của nếu và chỉ nếu một hoặc nhiều giao tác chứa . Tập tất cả những sự kiện đồng xuất hiện của được ký hiệu là .

**Định nghĩa 4:** Cho itemset và , độ đếm đồng xuất hiện của và , được ký hiệu là , được định nghĩa là độ hổ trợ của .

**Định nghĩa 5: (Top-k sự kiện đồng xuất hiện)** Cho itemset , một sự kiện được gọi là Top-k sự kiện đồng xuất hiện của nếu có ít hơn sự kiện mà độ điếm đồng xuất hiện của chúng lớn hơn .

**Phát biểu bài toán:** Cho một cơ sở dữ liệu giao tác , một itemset , và số mong muốn, bài toán tìm tập top-k những sự kiện đồng xuất hiện của là tìm sự kiện mà xảy ra phổ biến nhất với trong .

Giả định chúng ta muốn biết những sự kiện nào xảy ra phổ biến nhất với trong Bảng 1. Từ định nghĩa 4, ta biết rằng bởi vì chỉ chứa . Tương tự, ta có , , và . Nếu ngưỡng được thiết lập là 1, thì top-1 sự kiện đồng xuất hiện của là . Nếu , thì top-2 những sự kiện đồng xuất hiện của là và .

# Thuật toán (Navive Hunting algorithm)

Cho một itemset và cơ sở dữ liệu giao tác , quét tất cả các giao tác trong . Nếu một giao tác chứa , tăng số lượng đồng xuất hiện của mỗi sự kiện , không chứa trong trong giao tác, lên 1. Bằng cách duyệt tất cả các giao tác, có được số lượng đồng xuất hiện của tất cả các sự kiện đồng xuất hiện của . Bằng việc sắp xếp giảm dần tất cả các sự kiện đồng xuất hiện theo số lượng đồng xuất hiện, tìm thấy tất cả top-k sự kiện đồng xuất hiện của .

|  |
| --- |
| **Thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: , Top-k sự kiện đồng xuất hiện của  1: **foreach** **do**  2: **if** **then**  3: **foreach** **do**  4: **if** is fist visited **then**  5:  6: **else**  7:  8: is one of top-k biggest elements of  9: **return** |

# Thuật toán (Naive Hunting algorithm with Inverted list index)

Rỏ ràng, không hiệu quả khi nó quét tất cả giao tác trong cơ sở dữ liệu mà trong đó có nhiều giao tác không chứa itemset dẫn đến nó phải quét những giao tác không mong đợi. Để loại bỏ việc duyệt những giao tác không mong đợi, đầu tiên sẽ tìm tất cả các giao tác chứa itemset bằng cách sử dụng Tid-set[37], một cấu trúc dữ liệu được tổ chức từ danh sách được đảo ngược.Tid-set của một itemset là tập TIDs của tất cả các giao tác có chứa itemset đó. Như đã phát biểu trong [37], Tid-set có thể được tính toán bằng cách giao tất cả các Tid-set của những sự kiện mà được chứa bởi itemset đó. Khi đó, tìm ra top-k sự kiện đồng xuất hiện trong một cơ sở dữ liệu được hoạch định, là tập tất cả các giao tác chứa , như đã làm. Bởi vì chỉ tìm top-k sự kiện đồng xuất hiện trong một cơ sở dữ liệu nhỏ hơn, nên nó sẽ hiệu quả hơn .

|  |
| --- |
| **Giải thuật của thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: **foreach** **do**  2: **foreach** **do**  3: **if** is first visited **then**  4: TID\_list()  5: **else**  6: TID\_list() TID\_list()  7: TID\_list() TID\_list(*first-item*  8: **foreach** *first-item* **do**  9: TID\_list() TID\_list() TID\_list();  10: TID\_list()}  11: chạy thuật toán nhưng thay bằng ; |

# Cấu trúc BitTable

Bittable là cấu trúc dữ liệu dạng bảng mà mỗi bit biểu diễn một sự kiện xảy ra hay không. Nếu giá trị là 1 có nghĩa là sự kiện có xảy ra. Gọi cơ sở dữ liệu giao tác là ta ký hiệu bảng BitTable theo chiều ngang là và bảng BitTable theo chiều dọc là . Ví dụ ta có thể chuyển Bảng 1 từ cơ sở dữ liệu giao tác thành cấu trúc bittable như sau:

Bảng 2 BitTable theo chiều ngang

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| **1** | *0* | *1* | *0* | *0* | *0* | *1* | *1* |
| **2** | *1* | *1* | *1* | *0* | *0* | *1* | *0* |
| **3** | *1* | *0* | *1* | *1* | *0* | *1* | *0* |
| **4** | *0* | *1* | *1* | *0* | *1* | *0* | *0* |
| **5** | *1* | *0* | *1* | *1* | *1* | *1* | *0* |

Đối với bảng BitTable theo chiều ngang, mỗi cột tương ứng với một sự kiện, mỗi dòng tương ứng với một giao tác. Khi đó, nếu ta có một cơ sở dữ liệu giao tác với sự kiện và giao tác thì ta sẽ có một bảng BitTable với kích thước . Bảng BitTable theo chiều dọc cũng tương tự nhưng đổi cột thành dòng.

Bảng 3 BitTable theo chiều dọc

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **a** | *0* | *1* | *1* | *0* | *1* |
| **b** | *1* | *1* | *0* | *1* | *0* |
| **c** | *0* | *1* | *1* | *1* | *1* |
| **d** | *0* | *0* | *1* | *0* | *1* |
| **e** | *0* | *0* | *0* | *1* | *1* |
| **f** | *1* | *1* | *1* | *0* | *1* |
| **g** | *1* | *0* | *0* | *0* | *0* |

# Khai thác top-k đồng xuất hiện sử dụng cấu trúc bittable

# Thuật toán và

Tương tự như , nhưng khác ở chổ là xử lý trên cơ sở dữ liệu giao tác dạng text , còn xử lý trên bảng BitTable dạng ngang . cũng duyệt từng dòng của bảng BitTable để đếm số lần đồng xuất hiện của tất cả các sự kiện. Ví dụ đầu vào của là Bảng 1 thì của là Bảng 2.

Do mỗi dòng là một giao tác dạng bit, nên để kiểm tra itemset có được chứa trong giao tác hay không, chỉ cần thực hiện phép để kiểm tra. Ví dụ, xét bảng BitTable Bảng 2, để kiểm tra itemset có chứa trong giao tác hay không ta chỉ cần kiểm tra bit tại và bằng cách đơn giản là thực hiện phép AND giữa chuỗi bit 1110010 với 1010000 nếu kết quả bằng với 1010000 thì kết luận chứa trong . Tồng quát ta có .

|  |
| --- |
| **Thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: , Top-k sự kiện đồng xuất hiện của  1: Chuyển thành  2: Tạo từ  3: **foreach** **do**  4: **if** **then**  5: **foreach** **do**  6: **if** **then**  **7:**  8: **if** **then**  9:  10: **else**  11:  12: is one of top-k biggest elements of  13: **return** |

Rỏ ràng, không hiệu quả khi nó quét tất cả các dòng giao tác không mong đợi(không chứa itemset ). Để loại trừ điều đó, sử dụng Tid-set[37] tương tự như để xây dựng một cơ sở dữ liệu chỉ bao gồm những giao tác có chứa itemset . sẽ xây dựng trước, sau đó sẽ chuyển thành bảng BitTable theo chiều ngang để xử lý.

|  |
| --- |
| **Giải thuật của thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: **foreach** **do**  2: **foreach** **do**  3: **if** is first visited **then**  4: TID\_list()  5: **else**  6: TID\_list() TID\_list()  7: TID\_list() TID\_list(*first-item*  8: **foreach** *first-item* **do**  9: TID\_list() TID\_list() TID\_list();  10: TID\_list()}  11: chạy thuật toán nhưng thay bằng ; |

# Thuật toán (Bittable-based with Inverted list index in Vertical)

xử lý dựa vào bảng BitTable theo chiều dọc thay vì theo chiều ngang giống và . Đồng thời cũng loại bỏ những giao tác không chứa trước khi xử lý. Ví dụ, đối với cơ sở dữ liệu Bảng 1, cho itemset . sẽ loại bỏ những giao tác không chứa trước, khi đó ta sẽ có như bảng sau:

Bảng 4 cơ sở dữ liệu sau khi loại bỏ các giao tác không chứa

|  |  |
| --- | --- |
| TID | Items |
| 2 |  |
| 3 |  |
| 5 |  |

Sau đó sẽ chuyển Bảng dữ liệu trên thành BitTable dạng dọc

Bảng 5 bảng BitTable dạng dọc của tất cả những giao tác đều chứa

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **3** | **5** |
| **a** | *1* | *1* | *1* |
| **b** | *1* | *0* | *0* |
| **c** | *1* | *1* | *1* |
| **d** | *0* | *1* | *1* |
| **e** | *0* | *0* | *1* |
| **f** | *1* | *1* | *1* |
| **g** | *0* | *0* | *0* |

Tiếp theo loại bỏ những dòng dữ liệu của những sự kiện chứa trong itemset là và

Bảng 6 bảng BitTable mà sẽ xử lý

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **3** | **5** |
| **b** | *1* | *0* | *0* |
| **d** | *0* | *1* | *1* |
| **e** | *0* | *0* | *1* |
| **f** | *1* | *1* | *1* |
| **g** | *0* | *0* | *0* |

Để đếm số lần đồng xuất hiện của mỗi sự kiện, BTIV chỉ việc đếm số bit được set là 1 trong mãng bit tương ứng mà thôi. Rỏ ràng việc đếm số lượng bit 1 trong một mãng bit sẽ đơn giản và nhanh hơn rất nhiều so với việc duyệt từng bit để đếm số lượng đồng xuất hiện của mỗi sự kiện.

|  |
| --- |
| **Thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: Xây dựng từ  2: Chuyển thành bảng BitTable theo chiều dọc  3: **foreach** **do**  4: loại bỏ  5: **foreach** **do**  6:  7: is one of top-k biggest elements of  8: **return** |

# Kết quả đạt được

Trong phần này, chúng ta đánh giá hiệu suất của 3 thuật toán. Thử nghiệm được thực hiện trên một laptop cá nhân với cấu hình 4G Ram và bộ vi xử lý Intel Core i5-4200U 1.6 GHz. Tất cả code được hiện thực bằng ngôn ngữ C# và chạy trên hệ điều hành Win 8 64bit.

# Cơ sở dữ liệu

Hai cơ sở dữ liệu thực tế và hai cơ sở dữ liệu tổng hợp được dùng để chạy thử nghiệm. Bảng X thể hiện những thông số của các cơ sở dữ liệu. Cột cuối cùng của bảng X, chúng ta chia số lượng giao tác cho số lượng sự kiện để đánh giả tỷ trọng của cơ sở dữ liệu. Tỷ trọng càng lớn có nghĩa là cơ sở dữ liệu càng dày đặc.

Connect và Accidents là hai cơ sở dữ liệu thực tế. Chúng thường được sử dụng trong những công trình nghiên cứu trước về khai thác tập sự kiện phổ biến. Để kiểm tra các thuật toán trên cơ sở dữ liệu lớn, chúng ta tạo những cơ sở dữ liệu tổng hợp bằng chương trình IBM Quest Synthetic Data Generator. Cơ sở dữ liệu tổng hợp được đặc tên là Syn\_data1 và Syn\_data2 được sử dụng trong thử nghiệm. Để tạo ra Syn\_data1, kích thước trung bình mỗi giao tác, kích thước tối đa trung bình của tập sự kiện phổ biến tiềm năng, và độ tương quan giữa các mẫu được thiết lập lần lượt là 40,30 và 0.25, trong khi đó số giao tác, số sự kiện khác nhau, và số mẫu thì được thiết lập lần lượt là 1000K, 1K và 10. Những tham số để tạo Syn\_data2 cũng tương tự nhưng số mẫu thì được thiết lập là 100, để làm cho Syn\_data2 thưa thớt hơn Syn\_data1.

Bên dưới đây, chúng ta sẽ trình bày và phân tích các kết quả thực nghiệm trên nhiều phương diện khác nahu bao gồm thời gian tiền xữ lý, thời gian xữ lý và khả năng mở rộng.

# Thời gian tiền xữ lý

# Thời gian xử lý trên cơ sở dữ liệu Connect

# Thời gian xử lý trên cơ sở dữ liệu Accidents

# Kết luận và hướng phát triển