**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO**

TÌM HIỂU VỀ KHAI PHÁ LUẬT KẾT HỢP KHÔNG DƯ THỪA VÀ SO SÁNH 2 GIẢI THUẬT TOP-K RULES VÀ GIẢI THUẬT TNR

**Học phần: Seminar chuyên đề**

**Nhóm 2**

Trà Anh Vỹ - 3119410505

Vũ Quý Thiện - 3119560069

**GVHD**: Nguyễn Quốc Huy

**PHỤ LỤC**

**LỜI** **MỞ** **ĐẦU**………………………………………………………….3

PHẦN 1: KHAI PHÁ DỮ LIỆU và khai phá luật kết hợp………………………………………….………………………….4

1. phát triển tri thức…………………………………………………………………..4
2. Quá trình phát triển thi thức……………………………………………………….5
3. Khai phá dữ liệu……………………………………………………………………6
4. Mục đích của việc khai phá dữ liệu………………………………………………..6
5. Các ứng dụng của việc khai phá dữ liệu…………………………………………...6

PHẦN 2: KHAI PHÁ LUẬT KẾT HỢP……………………………..7

1. lý thuyết cơ bản…………………………………………………………………..7
2. Điều kiện thu luật kết hợp……………………………………………………..….8
3. Một số giải thuật và lý do chọn đề tài……………………………………………..9

PHẦN 3: GIẢI THUẬT KHAI PHÁ LUẬT KẾT HỢP KHÔNG DƯ THỪA…………………………………………………………………11

1. Khái niệm luật dư thừa…………………………………………………………..11
2. Giải thuật Top-K Rules
3. Giải thuật TNR…………………………………………………………………..18

phần 4: ỨNG DỤNG VÀ SO SÁNH 2 GIẢI THUẬT VỚI CÁC DATASET VÀ ĐƯA RA KẾT LUẬN………………………………22

LỜI MỞ ĐẦU

Với sự phát triển của công nghệ thông tin thì khối lượng dữ liệu lưu trữ ngày càng lớn, và giữa những lượng dữ liệu khổng lồ đó lại ẩn chứa một số thông tin được coi là chìa khóa dẫn đến thành công của mọi lĩnh vực từ hoạt động sản xuất đến kinh doanh. Việc khai phá, chiếc lọc thông tin ứng dụng vào cuộc sống của con người không chỉ dừng lại là một kĩ thuật đơn thuần, nó đòi hỏi sự ra đời của ngành khoa học mới: khoa học về phát hiện tri thức và khai phá dữ liệu (Knowledge Discovery and Data Mining - KDD).

Khai phá dữ liệu là ngành khoa học đang ngày được quan tâm nghiên cứu và phát triển do những ứng dụng thiết thực mà nó mang lại. Khai phá dữ liệu là phần cốt lõi của phát hiện tri thức, trong khai phá dữ liệu phát hiện các luật là một trong những nội dung cơ bản và phổ biến nhất. Các phương pháp phát hiện luật nhằm tìm ra sự phụ thuộc giữa các tính chất của các đối tượng hay các thuộc tính trong cơ sở dữ liệu.

Khai phá tri thức tiềm nằm trong tầm ngắm của nhiều nhà nghiên cứu trong lĩnh vực khai phá dữ liệu hiện nay. Với khối lượng dữ liệu ngày càng tăng, những thách thức đối với lĩnh vực này cũng ngày càng lớn hơn về cả độ phức tạp và quy mô. Trong đó, khai phá luật kết hợp không thừa là một hướng nghiên cứu mới, đặt ra yêu cầu thuật toán phải cho ra kết quả các luật kết hợp không thừa, giúp giảm bớt tập luật không cần thiết và tăng độ hiệu quả trong việc khai phá trên cơ sở dữ liệu lớn.

Trong nghiên cứu này, chúng tôi tập trung vào việc nghiên cứu hai thuật toán khai phá luật kết hợp không thừa là Top – K RuLe và TNR. Ngoài phân tích lý thuyết, các kết quả thực nghiệm cho thấy TNR là một lựa chọn tốt để khai phá luật kết hợp không thừa, với những chiến lược khai phá có thể có chi phí thấp hơn so với Top – K RuLe trong một số trường hợp, và ngược lại, có thể có chi phí cao hơn so với Top – K RuLe trong những trường hợp khác.

Phần 1: KHAI PHÁ DỮ LIỆU và khai phá luật kết hợp

* 1. Phát hiện tri thức VÀ KHAI PHÁ DỮ LIỆU (Knowledge Discovery & DATA MINING)
     1. Phát hiện tri thức

Chúng ta có thể xem tri thức như là các thông tin tích hợp, bao gồm các sự kiện và các mối quan hệ giữa chúng. Các mối quan hệ này có thể được hiểu ra được phát hiện hoặc cũng có thể được học. Nói cách khác tri thức có thể được coi là dữ liệu có độ trừu tượng và tổ chức cao.

Phát hiện tri thức trong các cơ sở dữ liệu là một quy trình nhận biết các mẫu hoặc các mô hình trong dữ liệu với các tính năng: hợp thức, mới, khả ích và có thể hiểu được. Còn khai phá dữ liệu là một bước trong quy trình phát hiện tri thức: gồm các thuật toán khai phá dữ liệu chuyên dùng dưới một số quy định về hiệu quả tính toán chấp nhận được để tìm các mẫu các mô hình trong dữ liệu. Nói một cách khác mục đích của phát hiện tri thức và khai phá dữ liệu chính là tìm ra các mẫu và các mô hình đang tồn tại trong cơ sở dữ liệu nhưng bị che khuất bởi hàng núi dữ liệu.

* + 1. Quá trình phát hiện tri thức
       1. Làm sạch dữ liệu (Data cleaning):

Là quá trình loại bỏ nhiễu - những bộ dữ liệu không bình thường, không tuân theo quy luật, nguyên tắc hay mô hình dữ liệu (còn gọi là các phần tử ngoài cuộc), và dữ liệu không nhất quán.

* + - 1. Tích hợp dữ liệu (Data intergation):

Dữ liệu có thể được thu thập từ nhiều nguồn khác nhau, hoặc có thể thu thập dữ liệu nhiều lần. Dữ liệu cuối của quá trình có có thể là kết quả của việc tổ hợp lại những lần thực hiện thu thập dữ liệu.

* + - 1. Lựa chọn dữ liệu (Data selection):

Kết quả đạt được của quá trình này là những dữ liệu thích hợp với nhiệm vụ phân tích được trích rút từ cơ sở dữ liệu.

* + - 1. Chuyển đổi dữ liệu (Data transformation):

Dữ liệu được chuyển đổi hay được hợp nhất về dạng thích hợp cho việc khai phá.

* + - 1. Khai phá dữ liệu (Data mining):

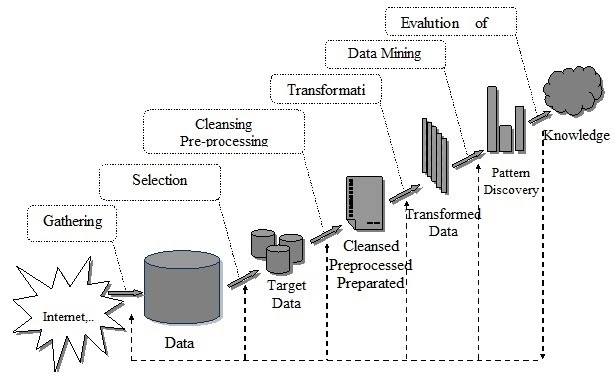
Đây là một tiến trình cốt yếu trong đó các phương pháp thông minh được áp dụng nhằm trích ra các mẫu dữ liệu.

* + - 1. Đánh giá mẫu (Pattern evaluation):

Dựa trên một số độ đo nào đó xác định lợi ích thực sự, độ quan trọng của các mẫu biểu diễn tri thức.

* + - 1. Biểu diễn tri thức (Knowledge presentation):

Ở giai đoạn này, các kĩ thuật biểu diễn và hiển thị được sử dụng để đưa tri thức đã lấy ra được cho người dùng.

****

* + 1. Khai phá dữ liệu

Ở một mức độ trừu tượng nhất định có thể định nghĩa về khai phá dữ liệu (*Data Mining)* là một quá trình tìm kiếm, phát hiện các tri thức mới, tiềm ẩn, hữu dụng trong CSDL lớn.

* + 1. Mục đích của việc khai phá dữ liệu
* Khai phá dữ liệu cung cấp những thông tin giúp hỗ trợ ra quyết định.
* Cung cấp những thông tin giúp dự báo: Ví dụ dự báo dân số thế giới căn cứ vào số liệu của dân số thế giới những năm trước đó.
* Có thể giúp khái quát dữ liệu.
  + 1. Các ứng dụng trong khai phá dữ liệu

Khai phá dữ liệu (KPDL) đang được áp dụng một cách rộng rãi trong nhiều lĩnh vực kinh doanh và đời sống khác nhau: marketing, tài chính, ngân hàng và bảo hiểm, khoa học, y tế, an ninh, internet… Rất nhiều tổ chức và công ty lớn trên thế giới đã áp dụng kĩ thuật khai phá dữ liệu vào các hoạt động sản xuất kinh doanh của mình và thu được những lợi ích to lớn. Các công ty phần mềm lớn trên thế giới cũng rất quan tâm và chú trọng tới việc nghiên cứu và phát triển kĩ thuật khai phá dữ liệu: Oracle tích hợp các công cụ khai phá dữ liệu vào bộ Oracle9i, IBM đã đi tiên phong trong việc phát triển các ứng dụng khai phá dữ liệu với các ứng dụng như Intelligence Miner ...

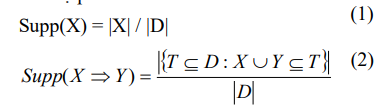
1. KHAI PHÁ LUẬT KẾT HỢP
   * 1. Lý thuyết cơ bản

Cho D = {t1, t2, …, tm} là cơ sở dữ liệu các giao dịch. Mỗi giao dịch ti bao gồm một tập n thuộc tính I = {i1, i2, …, in} và có một định danh duy nhất TID.

Một luật định nghĩa sự kéo theo có dạng X ⇒ Y trong đó X,Y ⊆ I và X ∩ Y = Ø. Trong đó, X gọi là phần mệnh đề điều kiện và Y gọi là mệnh đề kết quả của luật tương ứng.

***Độ hỗ trợ (Support) và độ tin cây (Confidence) là 2 tham số dùng để đo lường luật kết hợp.***

*Độ phổ biến (Support)*



*Độ tin cậy (Confident)*



* + 1. Điều kiện thu luật kết hợp

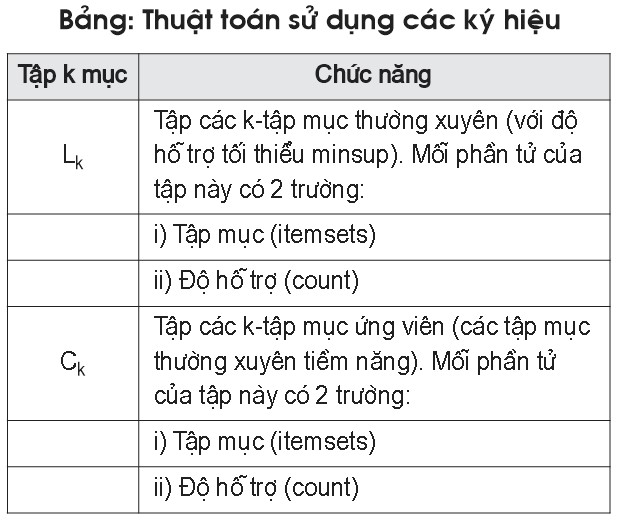
Để thu được các luật kết hợp, ta thường áp dụng 2 tiêu chí: **minimum support (min\_sup)** và **minimum confidence (min\_conf).** Các luật thỏa mãn có support và confidence thỏa mãn **(lớn hơn hoặc bằng)** cả Minimum support và Minimum confidence gọi là các luật mạnh **(Strong Rule).** Minimum support và Minimum confidence gọi là các giá trị ngưỡng **(threshold)** và phải xác định trước khi sinh các luật kết hợp.Một itemsets mà tần suất xuất hiện của nó **>= min\_sup** goi là **frequent itemsets.**

* + 1. Một số thuật toán khai phá luật kết hợp và lý do chọn khai phá luật kết hợp không dư thừa

1. Thuật toán Apriori

Apriori là thuật toán khai phá tập mục thường xuyên do R. Agrawal và R. Srikant đề xuất vào năm 1993. Thuật toán Apriori còn là nền tảng cho việc phát triển nhiều thuật toán khai phá tập mục thường xuyên khác về sau.

Giả sử các mục dữ liệu trong mỗi giao tác được lưu theo trật tự từ điển. Thuật toán sử dụng các ký hiệu sau: (Xem bảng).



Ý tưởng chính của thuật toán như sau: Sinh ra các tập mục ứng viên từ các tập mục thường xuyên ở bước trước, sử dụng kỹ thuật “tỉa” để bỏ đi những tập mục ứng viên không thoả mãn ngưỡng hỗ trợ cho trước. Cơ sở của kỹ thuật này là tính chất Apriori: Bất kỳ tập con nào của tập mục thường xuyên cũng phải là tập mục thường xuyên. Vì vậy, các tập mục ứng viên gồm k mục có thể được sinh ra bằng cách kết nối các tập mục thường xuyên có (k-1) mục và loại bỏ tập mục ứng viên nếu nó có chứa bất kỳ một tập con nào không phải là thường xuyên.

Thuật toán duyệt cơ sở dữ liệu nhiều lần. Mỗi lần duyệt, thuật toán thực hiện hai bước: bước kết nối và bước tỉa. Trong lần lặp thứ k, thuật toán nối hai (k-1) - tập mục để sinh ra k - tập mục, sử dụng tính chất Apriori để tỉa các tập ứng viên.

1. **Thuật toán FP\_growth**

Thuật toán Apriori có chi phí lớn nhưng lại kém hiệu quả. Để khắc phục nhược điểm này, J. Han, J Pei, Y. Yin và R. Mao đề xuất thuật toán FP-growth. Thuật toán FP-growth được xây dựng với 3 kỹ thuật chính:

(1) Nén dữ liệu thích hợp vào một cấu trúc cây gọi là cây FP-tree. Chỉ có các 1-tập mục (1-item) ở trong cây và các nút của cây được sắp xếp để các nút xuất hiện thường xuyên hơn có thể dễ dàng chia sẻ với các nút xuất hiện ít hơn.

(2) Thực hiện phương pháp khai phá phát triển (growth) từng đoạn dựa trên cây FP-tree gọi là phương pháp FP-growth.

(3) Kỹ thuật tìm kiếm được dùng ở đây là dựa vào sự phân chia, “chia để trị”, phân rã nhiệm vụ khai phá thành các nhiệm vụ nhỏ hơn.

Thuật toán FP-growth do nén toàn bộ CSDL lên một cấu trúc dữ liệu nhỏ hơn là cây FP-tree nên tránh được việc duyệt nhiều lần CSDL (thuật toán chỉ duyệt cơ sở dữ liệu 2 lần). Tiếp theo thuật toán khai phá cây bằng cách phát triển dần các mẫu mà không sinh các tập mục ứng viên, do đó tránh được khối lượng tính toán lớn. Phương pháp FP- growth đã chứng tỏ được tính hiệu quả của nó và có thể thực hiện khai phá cho cả các mẫu ngắn và dài, nhanh hơn thuật toán Apriori, luôn chỉ cần duyệt CSDL 2 lần.

Thuật toán FP- growth thực hiện như sau: Đầu tiên, thuật toán duyệt CSDL lần thứ nhất để tính độ hỗ trợ của từng mục (đếm số lần xuất hiện của từng mục). Tiếp đến, những mục không đủ độ hỗ trợ bị loại. Các mục còn lại được sắp theo thứ tự giảm dần của độ hỗ trợ (cũng tức là giảm dần theo số lần xuất hiện trong CSDL), ta nhận được danh sách L các mục đã sắp. Duyệt CSDL lần thứ hai, với mỗi giao tác t, loại các mục không đủ độ hỗ trợ, các mục còn lại theo thứ tự giống như xuất hiện trong L (tức là thứ tự giảm dần theo độ hỗ trợ) được cất vào cây FP-tree. Phần tiếp theo thuật toán khai phá tìm các mẫu thường xuyên trên cây FP-tree đã xây dựng mà không cần duyệt lại CSDL nữa.

1. **Các hướng chính mở rộng của Khai phá luật kết hợp và lý do chọn đề tài**

Lĩnh vực KPLKH cho đến nay đã được nghiên cứu và phát triển theo nhiều hướng khác nhau. Các hướng chính mở rộng là:

*- Luật kết hợp nhị phân*

*- Luật kết hợp có thuộc tính số và thuộc tính hạng mục*

*- Luật kết hợp tiếp cận theo hướng tập thô:*

*- Luật kết hợp nhiều mức*

*- Luật kết hợp mờ*

*- Khai phá luật kết hợp song song*

*- Khai phá luật kết hợp không dư thừa*

Và với mục đích nghiên cứu của bài báo chỉ ra một vấn đề của khai phá luật kết hợp là tìm tất cả các luật kết hợp trong một cơ sở dữ liệu thỏa tính chất: không ít hơn ngưỡng minsup mà người dùng đã định nghĩa và không thấp hơn minconf người dùng đưa ra. Mặc dù trước đó nhiều nghiên cứu đã được thực hiện trên khai phá luật kết hợp, một vấn đề quan trọng đã bị bỏ qua là cách người dùng nên chọn **minsup** và **minconf** để tạo ra một số lượng các luật như mong muốn.

Đây là một vấn đề quan trọng vì trong thực tế người sử dụng có thông tin (thời gian và không gian lưu trữ) giới hạn cho việc phân tích 28 kết quả, do đó người ta thường chỉ quan tâm đến phát hiện ra một số lượng nhất định các luật, việc điều chỉnh các tham số này là mất nhiều thời gian. Tùy thuộc vào sự lựa chọn các ngưỡng, các thuật toán hiện tại có thể rất chậm và tạo ra một số lượng rất lớn các kết quả hoặc không có hay quá ít kết quả, bỏ qua các thông tin có giá trị. Và đó là nhược điểm của các thuật toán được nêu ra trước đó, vì chúng chỉ có thể sàng ra tất cả, chỉ không loại bỏ những luật dư thừa không cần thiết. Chúng em quyết định nghiên cứu về hướng phát triển khai phá kết hợp không dư thừa để tối ưu trong tìm các luật kết hợp có giá trị mà giảm tải dữ liệu nhận về.

1. GIẢI THUẬT KHAI PHÁ LUẬT KẾT HỢP KHÔNG DƯ THỪA
2. Khái niệm luật dư thừa

Khai phá luật kết hợp bao gồm phát hiện mối liên quan giữa các tập mục trong các giao dịch. Nó là một trong những nhiệm vụ khai phá dữ liệu quan trọng nhất. Nó đã được tích hợp trong nhiều phần mềm khai phá dữ liệu thương mại và có nhiều ứng dụng đã được sử dụng trong thực tế.

Vấn đề khai phá luật kết hợp được quy định như sau. Cho i = {a1, a2, ... An} là một tập hợp hữu hạn các mặt hàng. Một cơ sở dữ liệu giao dịch là một tập hợp các giao dịch t = {t1, t2 ... Tm} trong đó mỗi giao dịch tj ⊆ i (1 ≤ j ≤ m) là đại diện cho một tập hợp các mặt hàng được mua bởi một khách hàng tại một thời điểm nhất định. Một tập phổ biến là một tập hợp các mặt hàng x ⊆ i. Một luật kết hợp x → y là một mối quan hệ giữa hai tập phổ biến x, y sao cho x, y ⊆ i và x ∩ y = ø. Độ hỗ trợ của một luật x → y được định nghĩa là sup (x → y) = sup (x ∪ y) / | t |. Độ tin cậy của một luật x → y được định nghĩa là conf (x → y) = sup (x ∪ y) / sup (x). Vấn đề của khai phá luật kết hợp là tìm tất cả các luật kết hợp trong một cơ sở dữ liệu thỏa tính chất: không ít hơn ngưỡng minsup mà người d ng đã định nghĩa và không thấp hơn độ tin cậy người d ng đưa ra (ngưỡng minconf). Ví dụ, bảng dưới cho thấy một cơ sở dữ liệu giao dịch (trái) và một số luật kết hợp tìm thấy cho minsup = 0,5 và minconf = 0,5 (bên phải).

Vấn đề dư thừa trong khai phá luật kết hợp đã được nghiên cứu rộng rãi và định nghĩa khác nhau về sự dư thừa của luật đã đề xuất. Tuy nhiên, nó vẫn là một thách thức mở để kết hợp các ý tưởng nhằm khai phá một bộ luật không dư thừa. Ý tưởng ở đây là đề xuất một thuật toán hiệu quả nhằm khai phá các luật kết hợp không dư thừa. Lợi ích của một thuật toán như vậy sẽ tìm được ra một tập hợp nhỏ k luật không dư thừa đối với người sử dụng. Tuy nhiên, việc nghĩ ra một thuật toán top-k để khai phá k những luật như vậy là rất khó. Lý do là loại bỏ sự dư thừa không thể thực hiện sau khi quá trình 29 khai phá các luật kết hợp đã kết thúc, bởi vì nếu loại bỏ ở giai đoạn này nó sẽ cho kết quả ít hơn k luật. Quá trình loại bỏ sự dư thừa do đó phải được tích hợp trong quá trình khai phá.

Các bộ luật không dư thừa thu được có thể được so sánh dựa trên một số tiêu chính như kích thước nhỏ gọn, khả năng phục hồi các luật dư thừa dựa trên đặc tính của chúng (độ hỗ trợ và độ tin cậy), và ý nghĩa của chúng đối với người sử dụng. Khái niệm luật dư thừa trong nghiên cứu này, chúng tôi chọn thuật toán TNR và căn cứ vào việc dựa trên điều kiện tối thiểu và đạt hiệu quả tối đa các luật.

Khái niệm luật dư thừa: Một luật ra : x → y là dư thừa đối với một luật rb : x1 → y1 khi và chỉ khi conf (ra) = conf (rb) ∧ sup (ra) = sup (rb) ∧ x1 ⊆ x ∧ y ⊆ y1. Ví dụ. Hãy xem các luật kết hợp trình bày trong bảng 3.1b. Luật {a} → {c, f} là dư thừa đối với luật {a} → {c,e,f}. Tương tự như vậy ta cũng có, luật {a, b} → {e, f} là dư thừa so với luật {a} → {e,f}. 3.1.2

Khai phá top-k luật kết hợp không dư thừa là khai phá ra một tập l (l là tập kết quả) có chứa k luật kết hợp từ cơ sở dữ liệu giao dịch cho trước. Đối với mỗi luật ra ∈ l | conf (ra) ≥ minconf, không tồn tại một luật rb ∉ l | conf (rb) ≥ minconf ∧ sup (rb)> sup (ra), nếu không rb là dư thừa so với ra . Hơn nữa, không tồn tại rc ,rd ∈ l mà rc là dư thừa đối với với rd.

1. Giải thuật Top-K Rules

Giải thuật Top-K Rules là một trong những giải thuật được sử dụng để khai phá luật kết hợp từ cơ sở dữ liệu. Mục tiêu của giải thuật này là tìm ra top-K luật kết hợp phổ biến nhất trong cơ sở dữ liệu, trong đó K là một số nguyên dương cho trước. Giải thuật này được đề xuất bởi Bing Liu, Wynne Hsu và Yiming Ma vào năm 1998.

**Các bước chính của giải thuật Top-K Rules như sau:**

1. Tìm tập phổ biến các itemset, sử dụng các giải thuật khai phá tập phổ biến như Apriori, FP-Growth,...

2. Tìm các luật kết hợp phổ biến từ tập phổ biến các itemset thu được ở bước trên. 3. Sắp xếp các luật theo độ phổ biến giảm dần và lấy ra K luật có độ phổ biến cao nhất.

Một số đặc điểm của giải thuật Top-K Rules là:

- Tập phổ biến các itemset được sử dụng để tìm luật kết hợp là một tập con của tập phổ biến các itemset tìm được bằng các giải thuật khai phá tập phổ biến khác.

- Giải thuật này có thể tìm ra các luật kết hợp phổ biến trong thời gian ngắn và sử dụng ít bộ nhớ.

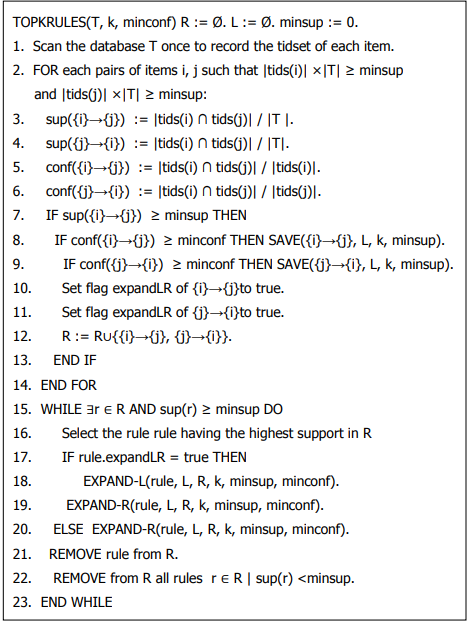
- Các luật kết hợp phổ biến được sắp xếp theo độ phổ biến giảm dần, giúp cho việc tìm kiếm các luật phổ biến dễ dàng hơn.

Tuy nhiên, giải thuật Top-K Rules cũng có một số hạn chế như:

- Không tìm được tất cả các luật phổ biến, chỉ tìm được top-K luật phổ biến nhất.

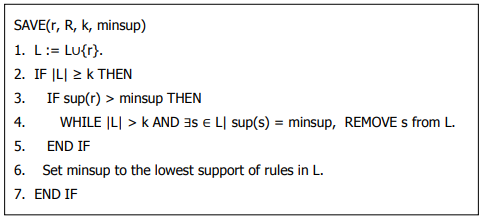
- Kết quả trả về phụ thuộc vào số lượng luật kết hợp phổ biến được tìm ra từ bước 2, do đó nếu số lượng luật phổ biến quá ít thì kết quả trả về có thể không chính xác.

- Việc xác định giá trị K phù hợp là một thách thức đối với giải thuật này.



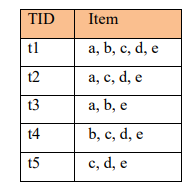
*Thuật toán Top-K Rules*

Thủ tục save(i→j,l,k,minsup) được mô tả chi tiết trong hình 3.2. Sau khi thêm luật r (i→j) vào L, nếu số lượng luật có trong L >=k và tồn tại luật s có sup(s) = minsup thì loại s khỏi l. Tính lại minsup = độ support thấp nhất của các luật có trong L.

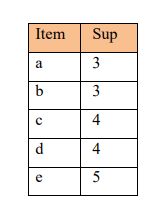


*Thủ tục SAVE*

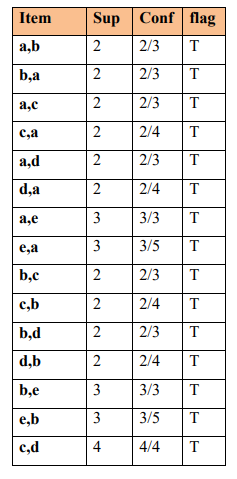
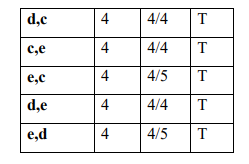
Để minh họa thuật toán, chúng ta xét ví dụ sau: giả sử có cơ sở dữ liệu giao dịch như bảng 3.2, cần khai phá k=10 luật kết hợp với minconf=70%. Các bước thực hiện được mô tả chi tiết.



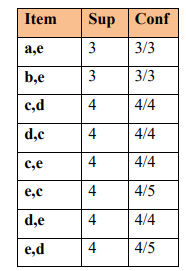
*Cơ sở dữ liệu giao dịch cho Top-K Rules*



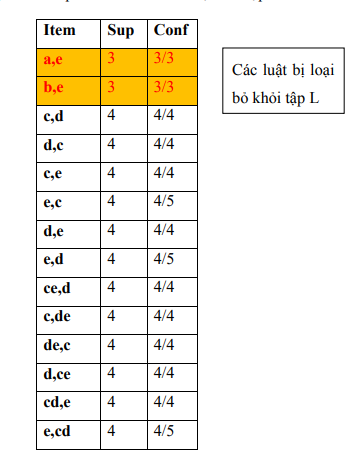
Bước 1: tính sup của từng item đơn

Bước 2: gán minsup=0; sinh các luật i→j và j → i thỏa điều kiện và đưa vào tập r



Bước 3: tìm các luật r có sup(r) >= minsup và conf (r)>= minconf để ghi vào tập l. Tập l gồm tập các luật thỏa điều kiện



Bước 4: trong r tìm luật có sup cao nhất để tiến hành thêm trái, thêm phải, nếu sau khi thêm mà luật mới thỏa minsup và minconf thì thêm luật mới vào l. Ví dụ, trong r chọn luật có sup cao nhất là c,d (c -> d) có sup=4. Nếu thêm trái và chọn luật thỏa điều kiện sẽ là ce,d (sup=4, conf=4/4), thêm ce,d vào l. Nếu thêm phải sẽ có c,de thỏa điều kiện (sup=4, conf=4/4) vậy thêm c,de vào l. Tới đây số luật đã có k=10 nên ta gán minsup=3 (sup nhỏ nhất có trong l). Thực hiện lần lượt với các luật còn lại trong bảng 3.4 có sup=4, khi số lượng k>=10 ta tìm và loại bỏ các luật có sup=minsup khỏi l. Gán lại minsup=4. Tiếp tục thực hiện ta có kết quả

Nhận xét giải thuật:

Thuật toán Top-K Rules là một thuật toán khai phá luật kết hợp không thừa, với mục đích tìm ra các luật kết hợp tần suất cao nhất và có ý nghĩa trong cơ sở dữ liệu. Các luật thu được từ thuật toán Top-K Rules đều có giá trị phân lớp cao và tối ưu về mặt chất lượng.

Một điểm mạnh của thuật toán Top-K Rules đó là tốc độ thực thi nhanh và hiệu quả khi khai phá trên cơ sở dữ liệu lớn. Thuật toán này sử dụng một số kỹ thuật tối ưu để giảm bớt số lượng luật không cần thiết và giảm độ phức tạp tính toán, giúp tăng tốc độ thực thi của thuật toán.

Tuy nhiên, thuật toán Top-K Rules cũng có một số hạn chế. Điều này liên quan đến việc thuật toán không thể tìm được tất cả các luật kết hợp có ý nghĩa trong cơ sở dữ liệu, do chỉ trả về top-K luật kết hợp tần suất cao nhất. Ngoài ra, trong một số trường hợp, thuật toán này có thể bỏ sót một số luật có ý nghĩa trong thực tế, do sự giới hạn của giá trị K được chọn.

Tóm lại, thuật toán Top-K Rules là một thuật toán khai phá luật kết hợp không thừa khá hiệu quả, với tốc độ thực thi nhanh và kết quả thu được tối ưu. Tuy nhiên, việc chọn giá trị K phù hợp và sử dụng thuật toán kết hợp với các phương pháp khai phá khác có thể giúp tăng cường hiệu quả của thuật toán.

1. Giải thuật TNR

Giải thuật TNR là một thuật toán khai phá luật kết hợp không dư thừa trong cơ sở dữ liệu. TNR được đề xuất để tìm kiếm các luật kết hợp với mức độ hỗ trợ (support) cao và mức độ tin cậy (confidence) cao. Thuật toán này sử dụng kỹ thuật "tilted window" để tạo ra các cửa sổ kết hợp và loại bỏ các luật có mức độ hỗ trợ thấp hơn ngưỡng được xác định trước đó.

Thuật toán TNR bao gồm các bước sau:

1. Tạo ra các cửa sổ kết hợp có kích thước k bằng cách di chuyển cửa sổ từ trái sang phải trên toàn bộ cơ sở dữ liệu.

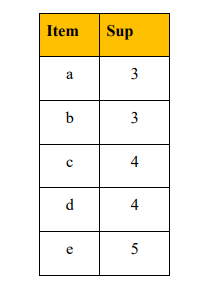
2. Tính toán mức độ hỗ trợ của các luật kết hợp thuộc cửa sổ và loại bỏ các luật có mức độ hỗ trợ thấp hơn ngưỡng được xác định trước đó.

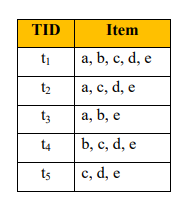
3. Sử dụng các luật kết hợp đã được lọc để tạo ra các cửa sổ mới với kích thước k+1.

4. Lặp lại các bước 2 và 3 cho đến khi không còn luật nào có thể tạo ra cửa sổ mới.

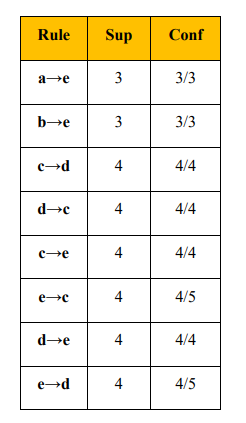
TNR có thể tìm kiếm các luật kết hợp có mức độ hỗ trợ và tin cậy cao, đồng thời giảm bớt số lượng luật dư thừa, tối ưu hóa bộ luật trích xuất và giảm thiểu thời gian xử lý. Tuy nhiên, TNR cần tiêu thụ nhiều bộ nhớ để lưu trữ các cửa sổ kết hợp.

*Cho bảng dữ liệu giao dịch như bảng 3.7 với điều kiện đầu vào là minconf = 70%, k =10 và yêu cầu tìm top-k luật không dư thừa. Các bước thực hiện được mô tả chi tiết như sau*

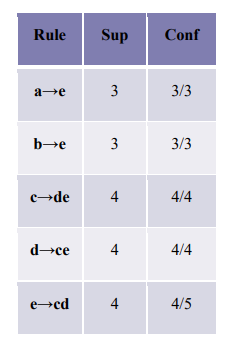
**

**

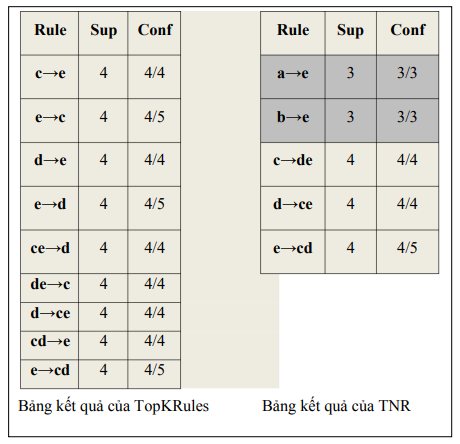
*Bước 1: tính sup của từng item đơn với minsup=0 ta có kết quả t*

**

*Bước 2: gán minsup=0, phát sinh các luật có dạng i→j và j→i. Nếu các luật này thỏa điều kiện lơn hơn minsup và minconf thì đưa vào tập r.*

**

*Bước 3: tìm kiếm – thay thế luật. Trong bước này thuật toán TNR sẽ thực hiện gần giống thuật toán Top-K Rules, điểm khác biệt là khi phát sinh ra luật ra thỏa điều kiện để thêm vào l thì sẽ kiểm tra trong l nếu có luật rb là dư thừa với ra thì loại rb ra khỏi l và nếu ra là dư thừa với rb thì ra không được thêm vào.*

**

*So sánh 2 bảng kết quả của hai thuật toán*

**Nhận xét giải thuật:**

Giải thuật TNR (Top Non-Redundant) là một phương pháp khai phá luật kết hợp từ cơ sở dữ liệu. Đây là một trong những giải thuật được đề xuất để giải quyết vấn đề luật dư thừa. TNR được thiết kế để tìm ra các luật kết hợp quan trọng nhất mà không có sự trùng lặp thông tin, giúp cho quá trình khai phá luật trở nên hiệu quả hơn.

TNR sử dụng phương pháp tìm kiếm trên cấu trúc cây FP-Tree để xác định các luật quan trọng và không dư thừa. Đầu tiên, TNR sử dụng giải thuật FP-Growth để xây dựng cây FP-Tree. Sau đó, TNR sử dụng thuật toán DFS (Depth First Search) để duyệt cây FP-Tree và tìm ra các luật quan trọng và không dư thừa. Các luật này được sắp xếp theo độ quan trọng và chỉ chọn ra K luật quan trọng nhất để đưa ra kết quả.

TNR là một giải thuật khá hiệu quả và cho kết quả khá tốt trong việc khai phá luật kết hợp. TNR đảm bảo rằng các luật thu được là quan trọng nhất và không dư thừa, do đó, kết quả trả về có tính chất tin cậy và chính xác cao. Tuy nhiên, nhược điểm của TNR là tốc độ thực thi khá chậm khi áp dụng cho các cơ sở dữ liệu lớn hoặc các tập dữ liệu có kích thước lớn.

Tóm lại, giải thuật TNR là một giải thuật khai phá luật kết hợp tốt và hiệu quả. TNR được thiết kế để giải quyết vấn đề luật dư thừa, giúp cho quá trình khai phá luật trở nên hiệu quả hơn. TNR có tính chất tin cậy và chính xác cao, tuy nhiên tốc độ thực thi khá chậm khi áp dụng cho các cơ sở dữ liệu lớn hoặc các tập dữ liệu có kích thước lớn.

1. ỨNG DỤNG VÀ SO SÁNH 2 GIẢI THUẬT VỚI CÁC DATASET VÀ ĐƯA RA KẾT LUẬN

Thuật toán Top-K Rules và TNR thí nghiệm được thực hiện trên 5 bộ dữ liệu

Máy tính: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx (8 CPUs), ~2.3GHz, 32768MB RAM, Windows 11

Ngôn ngữ lập trình Java sử dụng với trình biên dịch IntellDJ.

Mô tả dữ liệu: Dữ liệu sử dụng để chạy chương trình được download từ nguồn

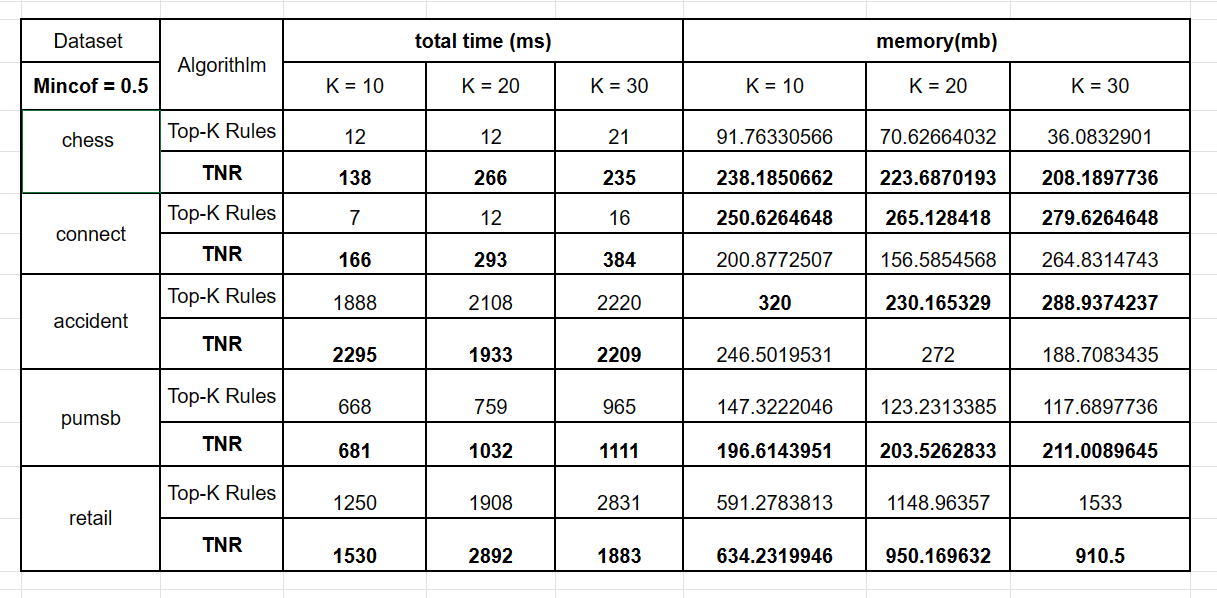
<http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php>.

Gồm chess, connect, retail, accident, pumsb

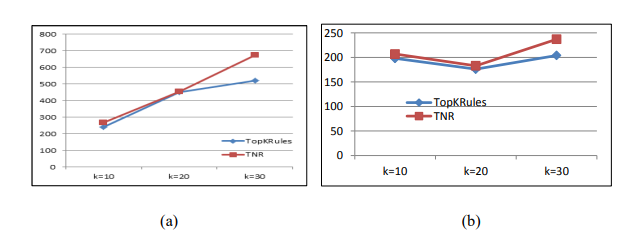
Chương trình thực nghiệm chương trình các thuật toán được lấy từ nguồn <http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php>.

Ngôn ngữ lập trình Java sử dụng với trình biên dịch IntellDJ.

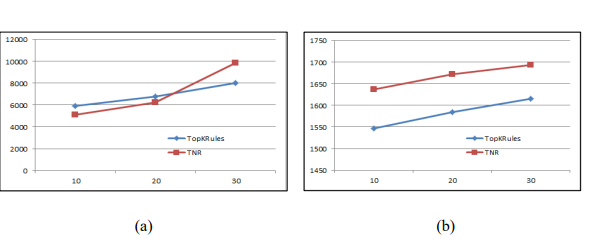
Để đảm bảo kết quả khách quan, chúng tôi chạy mỗi thí nghiệm 5 lần ứng với c ng một CSDL và hệ số k, ghi nhận kết quả của 5 lần chạy và lấy giá trị trung bình của chúng. Bảng kết quả thực nghiệm được ghi



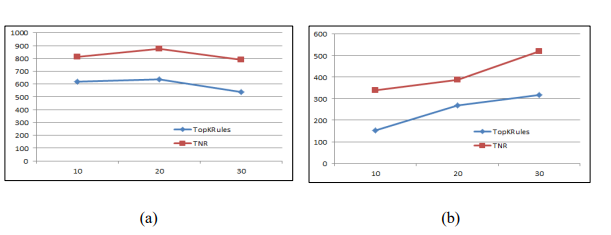
*Kết quả thực nghiệm với Top-K Rules và TNR (Trường hợp minConfident cố định)*



*So sánh thời gian thực hiện (a) và sử dụng bộ nhớ (b) của hai thuật toán Top-K Rules và TNR trên CSDL chess*

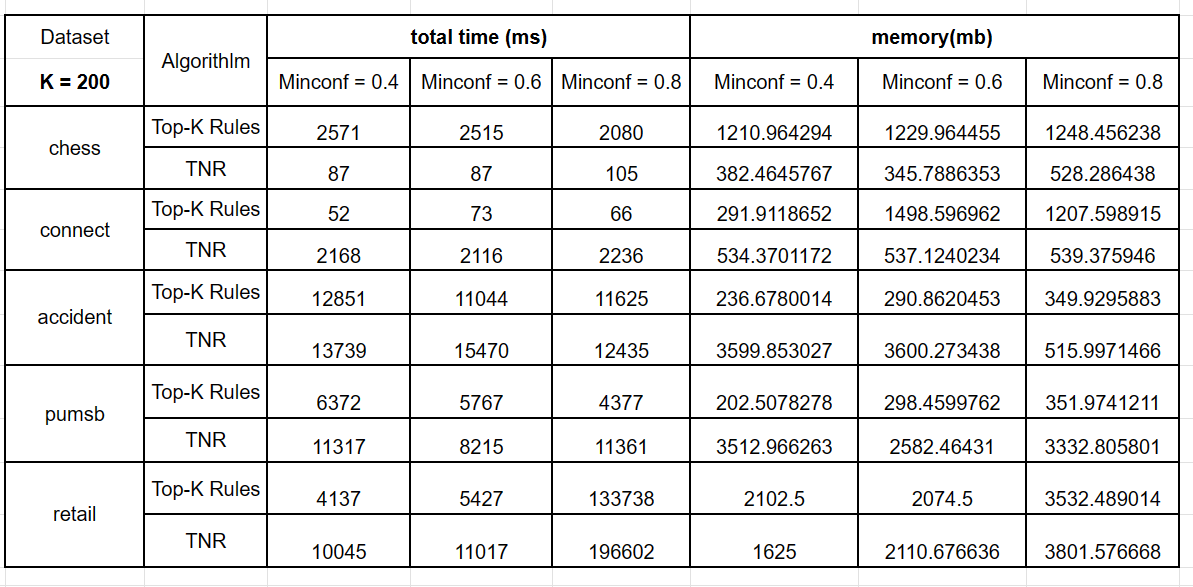
**

*So sánh thời gian thực hiện (a) và sử dụng bộ nhớ (b) của hai thuật toán Top-K Rules và TNR trên CSDL retail*

**

*So sánh thời gian thực hiện (a) và sử dụng bộ nhớ (b) của hai thuật toán Top-K Rules và TNR trên CSDL connect*

.

  
*Kết quả thực nghiệm với Top-K Rules và TNR (Trường hợp K cố định)*

Như vậy qua kết quả thực nghiệm trên 5 CSDL mẫu, chúng ta thấy thuật toán TNR cần thời gian và bộ nhớ cao hơn so với thuật toán Top-K Rules. Điều này có thể lý giải là do giải thuật tìm kiếm thay thế các luật dư thừa trong tập trung gian là mất nhiều thời gian và bộ nhớ hơn. Thay vào đó, kết quả các luật tìm được là không tồn tại luật dư thừa trong tập kết quả. Ngoài ra, tuy thuộc vào độ chính xác của thuật toán, nếu xảy ra tình trạng, hệ số delta thấp không đủ để thuật toán TNR chạy cho kết quả chính xác thì chúng ta phải thực hiện lại thuật toán với delta cao hơn, như vậy thời gian chạy sẽ cao hơn nhiều và tăng tỷ lệ thuận với độ lớn của số delta

Phần kết luận và hướng phát triển

Kết luận luận văn đã trình bày được phần lý thuyết về khai phá luật kết hợp và các thuật toán khai phá k luật kết hợp, luật kết hợp không dư thừa. Trong đó nêu rõ thuật toán khai phá k luật kết hợp, các ví dụ minh họa. Luật văn cũng nêu một số khái niệm về luật dư thừa và các chiến lược cải tiến trong thủ tục tìm kiếm và thay thế các luật dư thừa trong tập luật kết quả để thực sự thu được tập các luật mà trong đó không tồn tại luật dư thừa.

Do quá trình thay thế luật có thể phát sinh trường hợp mỗi luật ra được tìm thấy bởi chiến lược 2 đã chiếm một vị trí trong tập luật kết quả và như vậy sự hiện diện của nó trong tập kết quả có thể làm tăng biến minsup cục bộ (do chính minsup của luật ra). Nếu điều đó xảy ra, thuật toán có thể bị bỏ qua một số luật có độ hỗ trợ thấp hơn luật ra nhưng các luật bị bỏ đi này không là luật dư thừa. Như vậy, cần khắc phục tình trạng các luật không dư thừa có minsup thấp hơn luật dư thừa bị bỏ đi chúng tôi đề xuất sử dụng biến δ và một biến đếm cục bộ.

Chúng tôi đã thêm một biến đếm cục bộ. Biến này được tăng thêm 1 sau mỗi lần loại bỏ 1 luật từ tập kết quả của chiến lược 2. Sau đó, khi thuật toán kết thúc, biến đếm được so sánh với δ. Nếu giá trị biến đếm thấp hơn hoặc bằng δ, người d ng được thông báo rằng kết quả là chính xác. Ngược lại, người d ng được thông báo rằng kết quả có thể không chính xác. Trong trường hợp này, người d ng có thể chọn để chạy lại thuật toán với một giá trị δ cao hơn.

Kết quả thực nghiệm cho thấy rằng thuật toán tnr có thời gian chạy và chi phí bộ nhớ cao hơn so với thuật toán tìm k luật kết hợp (top-k rules). Điều này đã được chúng tôi giải thích ở phần thực nghiệm của chương 4. 52 hướng phát triển việc thay đổi số δ nhiều lần là một hạn chế lớn của thuật toán tnr. Trong tương lai chúng tôi sẽ nghiên cứu và đề xuất phương pháp chọn δ sao cho hiệu quả, để số lần phải chạy lại là ít nhất. Ngoài ra, thời gian thực hiện của thuật toán tnr còn khá cao so với thuật toán tìm k luật kết hợp.

Việc cải tiến thủ tục tìm kiếm và thay thế theo hướng tiếp cận các luật có tiềm năng là luật dư thừa thay vì phải quét tất cả các luật trong tập kết quả như hiện nay để tìm luật dư thừa. Nếu thực hiện được điều này, thuật toán mới sẽ cho kết quả chính xác với thời gian chạy ít hơn rất nhiều. Việc áp dụng thuật toán khai phá luật kết hợp với thời gian chạy ngắn hơn thay vì sử dụng thuật toán như hiện nay cũng sẽ được chúng tôi đưa vào hướng nghiên cứu tiếp theo

TÀI LIỆU THAM KHẢO

[1] Fournier-Viger, P., Wu, C.-W., Tseng, V. S. “Mining Top-K Association Rules”, Proc. 25th Canadian Conf. On Artificial Intelligence (Ai 2012), Springer, 2012, Pp. 61-73.

[2] Bay Vo, Bac Le “ A Frequent Closed Itemsets Lattice-Based Approach For Mining Minimal Non-Redundant Association Rules”, International Journal Of Database Theory And Application, Vol. 4, No 2, June 2011, Pp. 23 – 33.

[3] Philippe Fournier-Viger And Vincent S. Tseng. “Mining Top-K Non-Redundant Association Rules”. Foundations Of Intelligent Systems Lecture Notes In Computer Science Volume 7661, 2012, Pp 31-40.

[4] X.Wu, V.Kumar, J.Ross Quinlan, J.Ghosh, Q.Yang, H.Motoda, G.J. Mclachlan, A.Ng, B.Liu, P.S.Yu, Z.-H.Zhou, M.Steinbach, D.J.Hand, D. Steinberg (2008), Top 10 Algorithms In Data Mining, Knowl Inf Syst Volume14, Pp. 1–37