Khai thác luật kết hợp từ các tập mục hữu ích cao

Nguyễn Thị Thuý Loan[1], Mai Hoàng Thắng[2]

[1]Đại học Nguyễn Tất Thành
[2]Công Ty TNHH Harvey Nash Việt Nam
nthithuyloan@gmail.com; mhthang.it@gmail.com

Tóm tắt

Trong kinh doanh, các doanh nghiệp đều có chung một mong muốn là làm thế nào để tăng doanh thu hay lợi nhuận. Ví dụ, các siêu thị thường phân tích hoạt động kinh doanh của mình để xem xét sản phẩm nào mang lại lợi nhuận cao cho siêu thị. Để thực hiện được việc này, cần khai thác tập hữu ích cao. Gần đây có nhiều công trình quan tâm đến lĩnh vực này, nhưng các công trình trên tốn nhiều thời gian và bộ nhớ sử dụng trong quá trình khai thác. Trong công trình này, nhóm tác giả đề xuất một thuật toán giúp tiết kiệm được thời gian và bộ nhớ trong quá trình khai thác.

Nhận 05.03.2018
Được duyệt 18.05.2018
Công bố 19.06.2018

Từ khóa

Khai thác dữ liệu, tập hữu ích cao, luật kết hợp.

1. Giới thiệu

Khai thác dữ liệu (KTDL) là một quá trình quan trọng trong khám phá tri thức, nó là quá trình mô tả và dự đoán dựa trên các thông tin, tri thức, dữ liệu đã được lưu trữ, và phân tích các dữ liệu để tìm ra các dạng thức hoặc kết hợp có tính lặp đi lặp lại và tạo thành qui luật, các qui luật này hỗ trợ trong việc ra quyết định trong các lĩnh vực như: khoa học, giáo dục, kinh doanh, v.v... KTDL còn là quá trình phát hiện các mô hình, các tổng kết khác nhau và các giá trị được lấy từ tập dữ liệu cho trước [1].

Phương pháp KTDL thường được chia thành hai nhóm chính như sau:

(i) Kỹ thuật KTDL mô tả: có nhiệm vụ mô tả về các tính chất hoặc các đặc tính chung của dữ liệu string hiện có. Các kỹ thuật này bao gồm: Phân cụm (*Clustering*), tóm tắt (*Summerization*), trực quan hóa (*Visualization*), phân tích sự phát triển và độ lệch (*Evolution and Deviation analyst*), khai phá luật kết hợp (*Association rules*), …

(ii) Kỹ thuật KTDL dự đoán: Có nhiệm vụ đưa ra các dự đoán dựa vào các suy diễn trên dữ liệu hiện thời. Các kỹ thuật này gồm có: Phân lớp (*Classifacation*), hồi quy (*regession*), …. Tuy nhiên, chỉ có một số phương pháp thông dụng nhất là: Phân cụm dữ liệu, phân lớp dữ liệu, phương pháp hồi quy, và khai phá luật kết hợp.

Khai thác tập mục hữu ích cao là bài toán mở rộng và tổng quát của khái thác tập phổ biến. Trong khai thác tập mục hữu ích cao, giá trị của item trong giao dịch được quan tâm nhiều nhất (như số lượng đã bán của mặt hàng), ngoài ra còn có bảng lợi ích cho biết lợi ích mang lại khi bán một đơn vị hàng đó. Lợi ích của một itemset là số đo lợi nhuận của itemset đó đóng góp trong CSDL, nó có thể là tổng lợi nhuận hay tổng chi phí của itemset. Khai thác tập mục hữu ích cao là khám phá ra tất cả các tập mục có lợi ích không nhỏ hơn ngưỡng phổ biến tối thiểu do người dùng qui định. Mục đích chính của các bài toán khai thác tập mục hữu ích cao là làm giảm thiểu kích thước của tập ứng viên và làm đơn giản hóa quá trình tính toán độ hữu ích các tập mục từ đó giảm số lượng ứng viên cho tập mục hữu ích cao, giảm thời gian khai thác.

Cách tiếp cận đơn giản nhất cho bài toán khai thác tập mục hữu ích cao là liệt kê tất cả các tập mục từ CSDL giao dịch theo nguyên lý vét cạn, cách tiếp cận này sẽ gặp phải vấn đề về thời gian, không gian khi tìm kiếm quá lớn và nhất là khi CSDL chứa nhiều giao dịch hoặc ngưỡng *min-util* đặt ra quá thấp. Do đó, làm thế nào để tỉa bớt không gian tìm kiếm và tìm đủ tất cả tập mục hữu ích cao một cách hiệu quả là một thách thức lớn trong khai thác tính hữu ích.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 trình bày các nghiên cứu liên quan đến bài toán khai thác tập mục hữu ích cao, và khai thác luật kết hợp. Phần 3 trình bày thuật toán đóng góp bao gồm các định nghĩa liên quan và thuật toán đề xuất. Kết quả thực nghiệm được trình bày trong phần 4. Kết luận và hướng phát triển được trình bày trong phần 5.

2. Các công trình liên quan

Khai thác luật kết hợp truyền thống [2] chủ yếu dựa vào mô hình độ tin cậy – độ hỗ trợ. Theo đó, tất cả item trong cơ sở dữ liệu (CSDL) được xem xét như nhau. Tuy nhiên, trong

CSDL thực tế, mỗi item có trọng số riêng của nó. Do đó, có nhiều nghiên cứu liên quan đến mối quan hệ giữa trọng số của từng item với số lượng của nó. Khai thác tập mục hữu ích cao là một trong những chủ đề liên quan đến vấn đề này. Bài toán khai thác tập mục hữu ích cao giúp giải quyết vấn đề mà bài toán khai thác tập phổ biến không giải quyết được. Trong khai thác tập hữu ích cao (HUIM), các item có thể xuất hiện nhiều lần trong một giao dịch, mỗi item có một trong số (lợi nhuận, độ hữu ích…). Kết quả của khai thác tập mục hữu ích cao được ứng dụng để tìm ra itemsets trong cơ sở dữ liệu mang lại lợi nhuận cao.

Có rất nhiều thuật toán liên quan đã được đề xuất. Điển hình, Liu và các đồng sự (2005) đề xuất thuật toán Two-Phase với các khái niệm về độ hữu ích của giao dịch – Transaction Utility (TU) và trọng số hữu ích của giao dịch – Transaction Weighted Utility (TWU) để cải tiến không gian tìm kiếm khai thác tập hữu ích cao [3]. Bởi vì TWU của tập mục hữu ích thỏa mãn tính bao đóng giảm, do đó hoàn toàn có thể dựa vào TWU và sửa đổi các thuật toán khai thác tập phổ biến để khai thác tập hữu ích cao. Vì vậy, tác giả đã sửa đổi thuật toán Apriori để khai thác tập hữu ích cao. Thuật toán Two-Phase bao gồm hai giai đoạn chính như sau.

Giai đoạn 1: Tìm tất cả tập item có giá trị lợi ích lớn hơn giá trị ngưỡng do người dùng định nghĩa dựa trên trọng số hữu ích của giao dịch. Trong giai đoạn 1 chỉ có những kết hợp của những tập mục có trọng số giao dịch có độ hữu ích cao mới được thêm vào tập ứng viên trong suốt quá trình tìm kiếm thông minh trên mỗi mức. Tuy các tập item có độ lợi ích thấp có thể được đánh giá cao nhưng thuật toán lại không đánh giá thấp bất kỳ tập item nào.

Giai đoạn 2: Duyệt cơ sở dữ liệu để lọc ra các tập itemset có lợi ích cao từ tập lợi ích cao được tìm thấy trong giai đoạn 1. So với các thuật toán khai thác tập hữu ích cao hiện nay, thuật toán Two-Phase gặp vấn đề là một số lượng rất lớn các tập ứng viên được tạo ra nhưng hầu hết các ứng viên được sinh ra là có độ hữu ích không cao sau khi các giá trị hữu ích này được tính chính xác ở giai đoạn 2 của thuật toán. Ngoài ra, thuật toán thực hiện duyệt cơ sở dữ liệu nhiều lần sẽ gặp vấn đề về tốc độ xử lý nếu cơ sở dữ liệu có lượng giao dịch lớn.

Để giải quyết các vấn đề liên quan đến việc có nhiều tập ứng viên được sinh ra làm giảm năng suất thực hiện của thuật toán Two-Phase. Tseng và các đồng sự đã đề xuất thuật toán UP-Growth vào năm 2010 [4]. Thuật toán UP-Growth gồm hai bước chính. Bước 1, xây dựng cấu trúc cây Up-Tree. Bước 2, xác định các tập mục hữu ích cao từ các tập mục hữu ích cao tiềm năng (PHUIs). Trong giai đoạn đầu, thuật toán duyệt cơ sở dữ liệu để tính toán TWU cho từng item. Sau đó, ở giai đoạn hai, thuật toán duyệt cơ sở dữ liệu và loại bỏ những item có giá trị TWU nhỏ hơn ngưỡng độ hữu ích tối thiểu *min-util* ra khỏi giao dịch tương ứng. Mặc dù hướng tiếp cận này của thuật toán UP-Growth sinh ra ít ứng viên hơn trong giai đoạn 1. Việc duyệt CSDL gốc vẫn rất tốn thời gian do CSDL gốc quá lớn và vẫn còn chứa nhiều mục không triển vọng

Một cải tiến của thuật toán Up-Growth [4] được Tseng và các đồng sự đề xuất vào năm 2013 cũng nhằm mục đích khai thác các tập hữu ích cao, và được gọi tên là Up-Growth+ [5]. Thuật toán áp dụng các kỹ thuật cắt tỉa để rút gọn các tập các ứng viên. Sau khi tối ưu trên cây Up-Tree chúng ta sẽ có được tập các hữu ích cao tiềm năng (PHUIs) ít hơn so với Up-Growth. Thuật toán này được đánh giá là dễ cài đặt và có thời gian thực thi tốt hơn thuật toán Up-Growth vì chỉ thực hiện duyệt cơ sở dữ liệu hai lần.

Liu và Qu đã đề xuất thuật toán HUI-Miner (High Utility Itemset Miner) [6] để khai thác thác tập hữu ích cao sử dụng một cấu trúc mới, được gọi là danh sách lợi ích, để lưu trữ tất cả các thông tin hữu ích về một tập và tìm ra thông tin để cắt tỉa không gian tìm kiếm. Thuật toán HUI-Miner [6] được xem là thuật toán tốt nhất để khai thác tập hữu ích cao cho đến khi có sự xuất hiện của thuật toán FHM [7], một thuật toán khai thác tập hữu ích cao được đề xuất bởi Phillipe và các đồng sự vào năm 2014.

Khai thác luật kết hợp từ mẫu hữu ích cao

Bài toán khai thác luật kết hợp từ các mẫu hữu ích cao còn khá mới. Sahoo và các đồng sự đã khởi đầu nghiên cứu và đề xuất thuật toán khai thác luật kết hợp hữu ích cao [8] vào năm 2015. Thuật toán bao gồm ba giai đoạn chính, cụ thể như sau:

Giai đoạn 1: Khai thác các tập hữu ích cao đóng và các tập sinh.

Giai đoạn 2: Thực thi thuật toán HGB để tìm ra tập luật căn bản (high utility generic basic – HGB). Tập HGB được định nghĩa như sau: $HGB = \{R: g \rightarrow h \setminus g \mid h \in HUCI \wedge g \neq \emptyset, g \subset h, conf(R) \geq min - uconf \wedge \nexists g' \subset g \wedge conf(g' \rightarrow h \setminus g') \geq min - uconf$. Trong giai đoạn 2 này,

Giai đoạn 3: Thực thi thuật toán HAR để tìm ra tập kết quả tất cả các luật kết hợp hữu ích cao

Tên của thuật toán chung cho toàn bộ quá trình là HGB-HAR. Thuật toán HGB-HAR có khuyết điểm về mặt tính toán và tìm ra luật hợp lệ. Ngoài ra, luật sinh ra có thể bị trùng với luật đang có trong tập kết quả, do đó lãng phí thời gian tính toán. Vì vậy, thuật toán HGB-HAR chưa tối ưu về thời gian thực hiện.

3. Thuật toán đề xuất

3.1 Bài toán khai thác luật kết hợp hữu ích cao

Cho một cơ sở dữ liệu giao dịch *D*, ngưỡng độ hữu ích tối thiểu *min-util* và ngưỡng độ tin cậy hữu ích tối thiểu *min-uconf*, bài toán khai thác luật kết hợp hữu ích cao từ cơ sở dữ liệu *D* là tìm tất cả các luật có độ hữu ích lớn hơn hoặc bằng độ hữu ích tối thiểu *min-util* và có độ tin cậy hữu ích lớn hơn hoặc bằng độ tin cậy hữu ích tối thiểu.

3.2 Một số định nghĩa

Định nghĩa 1. Cho một tập mục hữu hạn chứa các mục $I = \{i_1, i_2, \ldots, i_m\}$, mỗi item i_p $(1 \le p \le m)$ được gắn với một lợi nhuận cố định, được ký hiệu $p(i_p)$. Một tập mục X gồm k mục phân biệt $\{i_1, i_2, \ldots, i_k\}$, trong đó $i_j \in I$, $1 \le j \le k$, k số phần tử trong tập mục X. Một cơ sở dữ liệu giao dịch $D = \{T_1, T_2, \ldots, T_n\}$ gồm tập các giao dịch T_d có một định danh id, được gọi là T_{id}. Mỗi item i_p trong mỗi giao dịch T_d được gắn kết với một trọng số được gọi là số lượng và được ký hiệu là $q(i_p, T_d)$, tương ứng với item i_p được mua.

Định nghĩa 2. Độ hữu ích của một item i trong một giao dịch T_d được ký hiệu là $u(i, T_q)$ và được định nghĩa bằng công thức $p(i) \times q(i, Td)$.

Định nghĩa 3. Độ hữu ích của một tập mục X trong giao dịch T_d được ký hiệu là $u(X,T_d)$ và được xác định bởi công thức:

$u(X, T_d) = \sum_{x_i \in X} u(x_i, T_d)$.

Định nghĩa 4. Độ hữu ích của một tập mục X trong cơ sở dữ liệu D được tính bằng tổng tất cả các độ hữu ích của X trong tất cả các giao dịch có chứa X.

$u(X) = \sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d)$.

Định nghĩa 5. Một tập mục X được xem là tập mục hữu ích cao (HUI) nếu X có độ hữu ích bằng hoặc lớn hơn giá trị hữu ích tối thiểu mà người dùng định nghĩa (*min-util*). Nếu tập mục X có độ hữu ích thấp hơn độ hữu ích tối thiểu thì X không phải là tập mục hữu ích cao, hay còn gọi là tập mục hữu ích thấp.

Định nghĩa 6. Một tập mục Y được gọi là tập bao đóng của tập mục X nếu không có tập cha nào của X chứa Y và có $supp(X) = supp(Y)$, ký hiệu là $\gamma(X)$. X được gọi là tập hữu ích đóng nếu $X = \gamma(X)$ và $u(X) \ge min\text{-}util$.

Định nghĩa 7. Một tập mục X được gọi là tập sinh hữu ích cao (HUI Generator) nếu X là tập mục hữu ích cao và không có tập con Z nào của X sao cho $supp(X) = supp(Z)$.

Định nghĩa 8. Độ hữu ích cục bộ của một item x_i trong tập mục X, ký hiệu $luv(x_i, X)$ và được tính bằng tổng độ hữu ích của x_i trong tất cả giao dịch có chứa X, được xác định bằng công thức sau:

$luv(x_i, X) = \sum_{X \subseteq t_d \wedge t_d \in D} u(x_i, t_d)$.

Định nghĩa 9. Với $X = \{x_1, x_2, \ldots, x_n\}$ là một tập mục n phần tử, mảng đơn vị độ hữu ích của X được ký hiệu $U(X) = \{u_1, u_2, \ldots, u_n\}$, trong đó $u_i = luv(x_i, X), i \in \{1,2,\ldots,n\}$.

Định nghĩa 10. Độ hữu ích cục bộ của tập mục X trong tập mục Y $(X \subseteq Y)$, ký hiệu là $luv(X, Y)$ và được định nghĩa bằng tổng các độ hữu ích cục bộ của tất cả item $x_i \in X$ trong Y. Công thức tính độ hữu ích cục bộ của tập mục X trong tập mục Y được biểu diễn như sau:

$luv(X, Y) = \sum_{x_i \in X \subseteq Y} luv(x_i, Y)$.

Định nghĩa 11. Luật kết hợp hữu ích R là một hàm biểu diễn mối quan hệ giữa hai tập hữu ích cao $X, Y \subseteq I$, được biểu diễn dưới dạng $X \to Y$. Độ tin cậy hữu ích của luật R, ký hiệu là $uconf(R)$, được xác định bằng công thức $(R) = \frac{luv(X,\ XY)}{u(X)}$.

$R: X \to Y$ được gọi là luật kết hợp hữu ích cao nếu giá trị của $uconf(R)$ lớn hơn hoặc bằng độ tin cậy hữu ích tối thiểu (*min-uconf*) do người dung định nghĩa. Ngược lại, R được gọi là luật kết hợp hữu ích thấp.

Tính chất 1. Cho $R1: X \to Y$, $R2: X \to Z$ $(Y \subset Z)$ là hai luật kết hợp trong mô hình độ tin cậy – hữu ích (utility-confidence framework), nếu $R1$ không phải là luật kết hợp hữu ích cao, thì $R2$ cũng không phải là luật kết hợp hữu ích cao.

Định nghĩa 12. Cho $R1: X1 \to Y1$ và $R2: X2 \to Y2$ là hai luật kết hợp hữu ích cao trong mô hình độ tin cậy – hữu ích. $R2$ được xác định là dư thừa so với $R1$ nếu $X2 \cup Y2 \subseteq X1 \cup Y1, R1.utility \ge R2.Utility, support(R1) = support(R2)$ và $X1 \subseteq X2, Y2 \subseteq Y1$, trong đó $Ri.utility$ là độ hữu ích của luật $Ri, i = 1,2$, và độ hỗ trợ của luật $R: X \to Y$ là $supp(X \cup Y)$.

3.3 Thuật toán

Thuật toán HUIL

Đầu vào: Tập HUIs được sắp xếp theo thứ tự phần tử tăng dần (*TableHUI*)

Đầu ra: dàn HUIL với nút gốc *rootNode*

```
BuildLattice()
1.  Set rootNode = Lattice Node of Empty Itemset;
2.  For j = 1 to TableHUI.Levels.Count do
3.      For each X in TableHUI.Levels[j] do
4.          Set rootNode.IsTraversed = False;
5.          For each childNode in rootNode.Children do
6.              If childNode.Itemset ⊂ X then
7.                  Set childNode.IsTraversed = False;
8.                  ResetLattice (childNode);
9.              End
10.         End
11.         InsertLattice (HUI, rootNode);
12.     End
13. End

ResetLattice (LatticeNode)
14. Set LatticeNode.IsTraversed = False;
15. For each child in LatticeNode.Children do
16.     ResetLattice (child);
17. End

InsertLattice (X, rootNode)
18. If rootNode.IsTraversed then
19.     return;
20. End
21. Set Flag = True, rootNode.IsTraversed = True;
22.     For each childNode in rootNode.Children do
23.         If childNode ⊂ X then
24.             Set Flag = False;
25.             InsertLattice (X, childNode);
26.         End
27.     End
28. If Flag = True then
29.     rootNode.Children.Add (X);
30. End
```

Hình 1: Thuật toán HUIL

Thuật toán xây dựng dàn từ các HUIs được thực hiện như sau:

Đầu tiên, thuật toán HUIL sẽ gọi hàm BuildLattice để xây dựng nút gốc cho dàn. Nút gốc là một nút rỗng không có chứa HUI, không có giá trị hữu ích và độ hỗ trợ.

Tiếp theo, thuật toán duyệt qua tất cả các HUIs theo thứ tự sắp xếp số phần tử tăng dần. Khi xét mỗi HUI, thuật toán sẽ khởi tạo lại giá trị của cờ IsTraversed cho nút gốc và các nút con.

Sau đó, thuật toán gọi hàm InsertLattice để thực hiện thêm HUI vào dàn. Trong hàm InsertLattice, cờ được sử dụng để xác định xem HUI đang xét $\{X\}$ có thể được thêm trực tiếp vào nút đang xét hay không. Nếu nút đang xét *rootNode* có các nút con $childNode$ sao cho $childNode \subset X$ (dòng 23), hàm InsertLattice sẽ được gọi đệ quy (dòng 25) để thêm nút $\{X\}$ vào dàn. Nếu không có nút con *childNode* nào sao cho $childNode \in rootNode.Children$ và $ChildNode \subset X$, X sẽ là nút con trực tiếp của nút đang xét *rootNode* (dòng 29).

4. Thực nghiệm

4.1 Môi trường thực nghiệm

Các thuật toán đề xuất được cài đặt và thực nghiệm trên môi trường có cấu hình như sau: Intel Core I7-7500U 2.5 GHz, Ram 16 GB, hệ điều hành Windows 10, phiên bản 64 bit. Công cụ dùng để phát triển thuật toán: Visual Studio 2015 Community, .Net framework 4.5, ngôn ngữ C#.

4.2 Cơ sở dữ liệu thực nghiệm

Các cơ sở dữ liệu dùng cho thực nghiệm là các cơ sở dữ liệu chuẩn được tải từ website mã nguồn mở SPMF phát triển bởi Philippe (http://www.philippe-fournier-viger.com/ spmf/ index.php?link=datasets.php). Các thuộc tính của cơ sở dữ liệu được mô tả trong Bảng 1.

Bảng 1. Thuộc tính của các cơ sở dữ liệu.

Tên	Số giao dịch	Số lượng items	Kích thước (MB)
Chess	3,196	75	0.63
Foodmart	4,141	1,559	0.17
Retail	88,162	16,470	6.42
Chainstore	1,112,949	46,086	79.2

4.3 Kết quả thực nghiệm

Thuật toán FHIM được đề xuất bởi Sahoo và các đồng sự [8] được dùng để khai thác các tập mục hữu ích cao từ các cơ sở dữ liệu được đề cập ở trên. Sau đó thuật toán được đề xuất sẽ được thực thi với các thông số đầu vào bao gồm các tập hữu ích cao, độ hữu ích tối thiểu *min-util*, độ tin cậy hữu ích tối thiểu *min-uconf*.

Bảng 2. Kết quả số luật kết hợp hữu ích cao trên các CSDL thực nghiệm.

CSDL	*min-util* (%)	#HUIs	#HARs (*min-uconf* = 60%)	#HARs (*min-uconf* = 70%)	#HARs (*min-uconf* = 80%)
Foodmart	0.03	54,928	3,099,516	3,098,322	3,098,176
	0.04	20,766	810,707	810,488	810,42
	0.05	2,266	105,805	105,785	105,740
	0.06	1,483	4,891	4,891	4,891
Chess	27.5	791	30,726	30,144	22,211
	28.0	493	14,287	14,197	11,512
	28.5	305	6,677	6,668	5,844
	29.0	176	2,893	2,893	2,701
Chainstore	0.005	12,347	718	439	342
	0.01	3,884	113	77	65
	0.02	1,165	15	12	11
	0.03	593	7	6	6
Retail	0.01	22,479	22,120	13,642	6,016
	0.02	7,375	6,725	3,827	1,472
	0.03	3,765	3,160	1,755	673
	0.04	2,272	1,873	1,033	397

Kết quả luật kết hợp hữu ích cao với độ tin cậy hữu ích tối thiểu 60% - 80% và các độ hữu ích tối thiểu tương ứng của từng cơ sở dữ liệu được liệt kê trong Bảng 2.

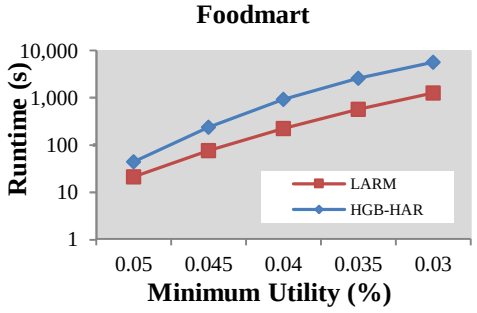
4.4 So sánh về thời gian

Thuật toán đề xuất LARM có thời gian thực thi tối ưu nhờ vào cải tiến không gian tìm kiếm thông qua việc áp dụng tính chất 1 đã đề cập ở trên. Kết quả là số cặp itemset cần xét để hình thành luật giảm.

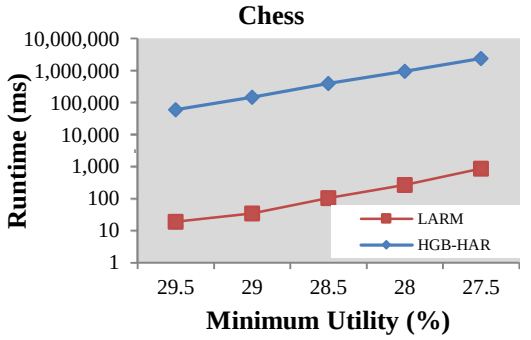
Trong phần tiếp theo của thực nghiệm, các đồ thị so sánh về thời gian thực thi sử dụng giữa hai thuật toán LARM và HGB-HAR sẽ được trình bày dưới dạng đồ thị sử dụng tỉ lệ thang logarit của 10. Một số ký hiệu cho các đường biểu diễn trên đồ thị cụ thể như sau.

LARM: biểu diễn cho thời gian thực thi để khai thác luật kết hợp hữu ích cao, bao gồm thời gian xây dựng dàn và thời gian rút trích luật.

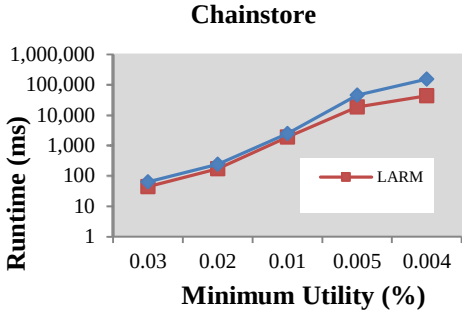
HGB-HAR: biểu diễn cho thời gian thực thi của thuật toán HGB-HAR



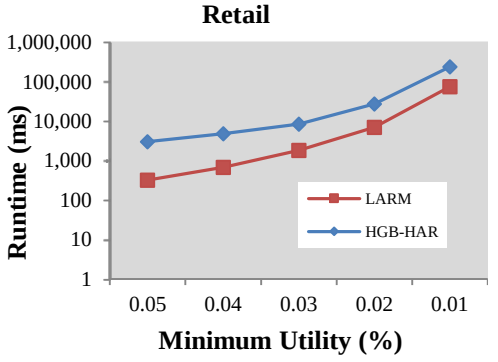
Hình 2. Thời gian thực thi trên CSDL Foodmart với *min-uconf* = 70%.



Hình 3. Thời gian thực thi trên CSDL Chess với *min-uconf=70%*



Hình 4. Thời gian thực thi trên CSDL Chainstore với *min-uconf* = 70%.



Hình 5. Thời gian thực thi trên CSDL Retail với *min-uconf* = 70%.

Kết quả từ Hình 2 đến Hình 5 có thể đánh giá được rằng thuật toán LARM là thuật toán có thời gian thực thi tối ưu. Bên cạnh đó, nếu không xét đến thời gian xây dựng dàn, thuật toán LARM sẽ sử dụng rất ít thời gian để tìm kết quả luật kết hợp hữu ích cao. Các kết quả thực nghiệm trên các CSDL đã chứng minh ưu thế của việc sử dụng dàn trong khai thác luật kết hợp, đặc biệt là luật kết hợp hữu ích cao.

5. Kết luận và hướng phát triển

5.1 Kết luận

Trong nghiên cứu này, tác giả sử dụng mô hình độ tin cậy hữu ích và lý thuyết dàn để khai thác luật kết hợp hữu ích cao nhằm khai thác mối quan hệ giữa các tập mục hữu ích cao. Nghiên cứu này là nghiên cứu đầu tiên áp dụng lý thuyết về dàn trong khai thác luật kết hợp hữu ích cao. Tác giả đã đề xuất thuật toán HUIL để xây dựng dàn gồm các tập mục hữu ích cao. Kết quả thực nghiệm trên một số cơ sở dữ liệu chuẩn cho thấy thuật toán đã đề xuất, LARM, có hiệu quả cao cả về thời gian thực thi và bộ nhớ sử dụng. Tính hiệu quả của thuật toán sẽ đóng góp rất lớn trong các hệ thống dự báo và ra quyết định.

Nghiên cứu này có thể được ứng dụng hiệu quả trong sản xuất kinh doanh, lập kế hoạch kinh doanh cũng như cuộc sống dựa vào đặc điểm và tính chất ứng dụng luật ứng với mỗi luật trong tập luật. Kết quả từ các luật kết hợp hữu ích cao sẽ mang lại kết quả hữu ích cho lãnh đạo trong khi hoạch định kế hoạch sản xuất, kinh doanh trong thời gian sắp tới, điển hình như xem xét các tập mặt hàng kết hợp với nhau mang lại lợi nhuận cao trong hoạt động kinh doanh bán lẻ, hoặc để xuất các chương trình khuyến mãi nhằm mang lại hiệu quả kinh doanh cao nhất.

5.2 Hướng phát triển

Bằng cách sử dụng thuật toán HUIL để xây dựng kiến trúc dàn các tập hữu ích cao, nghiên cứu này có thể mở rộng phát triển các thuật toán khai thác luật kết hợp hữu ích cao không dư thừa, ngoài ra, có thể phát triển thuật toán khai thác các tập đóng hữu ích cao (closed high utility itemsets) và tập sinh hữu ích cao (high utility generators). Bên cạnh đó, các độ đo thú vị [9], [10] có thể được nghiên cứu áp dụng vào các thuật toán đã đề xuất nhằm tăng thêm tính hiệu quả và khai thác thêm các thông tin hữu ích từ các cơ sở dữ liệu giao dịch.

Lời cảm ơn

Nghiên cứu này được tài trợ bởi Quỹ Phát triển khoa học và công nghệ NTTU trong đề tài mã số 2017.01.75

Tài liệu tham khảo

1. B. Ho, "Introduction to Knowledge Discovery and Data Mining," National Center for Natural Science and Technology, 1998.
2. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993, pp. 207-216.
3. Y. Liu, W. Liao, and A. Choudhary, "A Two-Phase algorithm for fast discovery of high utility itemsets.," in *Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, 2005, pp. 689-695.
4. S. V. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-Growth: an efficient algorithm for high utility itemset mining," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 253-262.
5. V.S. Tseng, C Wu, B Shie, and P.S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1772–1786, 2013.
6. M. Liu and J. Qu, "Mining high utility itemsets without candidate generation.," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 55-64.
7. P. Fournier-Viger, C. Wu, S. Zida, and V.S. Tseng, "Faster high utility itemset mining using estimated utility co-occurrence pruning," in *Proceedings 21st International Symposium on Methodologies for Intelligent Systems*, 2014, pp. 83-92.
8. J. Sahoo, A.K. Das, and A. Goswami, "An efficient approach for mining association rules from high utility itemsets," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5754-5778., 2015.
9. L. Nguyen, B. Vo, and T. Hong, "CARIM: An efficient algorithm for mining class association rules with interestingness measures," *The International Arab Journal of Information Technology*, vol. 12, no. 6A, pp. 627-634, 2015.
10. B. Vo and B. Le, "Interestingness for association rules: combination between lattice and hash tables," *Expert Systems with Applications* , vol. 38, no. 9, pp. 11630–11640, 2011.

Mining association rules from high utility itemsets

Nguyen Thi Thuy Loan[1], Mai Hoang Thang[2]
[1]Nguyen Tat Thanh University
[2]NashTech Global

Abstract Most companies focus on their profit growth within the business environment. For example, supermarkets often analyze sales activities to investigate which products bring the most revenue. In order to solve the problem, we need to mine high utility item sets. Recently, there have been many researches focus on this problem. However, these methods consume more time and memory usage. In this paper, we propose an algorithm for saving the mining time and memory usage during mining process.

Key words Data mining, high utility itemsets, association rules.