GIẢM CHIỀU RỘNG CÂY DỰ ĐOÁN NÉN VỚI GIẢI PHÁP BẢO TOÀN THÔNG TIN

● NGUYỄN THÔN DÃ

TÓM TẮT:

Bài báo này đề xuất một giải pháp giảm kích cỡ chiều rộng cây dự đoán nén (CPT - Compact Prediction Tree) với giải pháp không làm mất thông tin. Chiến lược của giải pháp là dựa vào phân tích chuỗi dự đoán để cắt tỉa không gian của cây dự đoán theo chiều rộng nhằm thu hẹp kích cỡ của cây dự đoán nén và kiểm tra việc bảo toàn thông tin dựa vào thuật toán nối bảo toàn. Điều này rất hữu ích trong việc dự đoán chuỗi tuần tự vì khi được giảm kích cỡ, việc truy xuất, tìm kiếm, dự đoán sẽ thuận lợi hơn và giảm chi phí thời gian và bộ nhớ khi áp dụng mô hình CPT hay CPT+ cho dự đoán chuỗi tuần tự...

Từ khóa: CPT, CPT+, bảo toàn thông tin, cây dự đoán nén.

1. Đặt vấn đề

Dự đoán chuỗi tuần tự được ứng dụng trên thực tế với nhiều lãnh vực khác nhau như trong dự đoán chuỗi protein, chuẩn đoán tim mạch (Rjeily, Badr, El Hassani, & Andres, 2017), dự đoán xu hướng thị trường chứng khoán (Ngân hàng) (Sun & Giles, 2001), dự đoán các sản phẩm (Kinh doanh), dự đoán thời tiết, ứng dụng trong giao thông (Bernard & Andritsos, 2019), dự đoán hiệu quả việc học của sinh viên (Previde, 2019)...

Việc thu hẹp không gian tìm kiếm và dự đoán rất quan trọng, Gueniche, Fournier-Viger, Raman, & Tseng, 2015, đã đề xuất phương pháp giảm chiều cao của cây dự đoán nén bằng hai chiến lược Frequent subsequence compression (FSC), Simple Branches Compression (SBC) Ý tưởng của chiến lược FSC là thay thế các chuỗi giống nhau bằng một nút biểu diễn cho các chuỗi giống nhau đó. Bên cạnh đó, chiến lược SBC là chiến lược thực hiện sau khi chiến lược FSC thực hiện xong. Ý tưởng của chiến lược SBC là thay thế các nhánh con đơn giản thành một nút. Hai chiến lược này đã giúp việc truy xuất và dự đoán trên cây CPT được hiệu quả hơn. Đây chính là 2 trong 3 đề xuất quan trọng của phương pháp CPT+ (cải tiến từ phương pháp CPT). Bài báo này đề xuất một giải pháp giảm kích cỡ cây theo chiều rộng để hỗ trợ việc dự đoán bằng CPT+ được hiệu quả hơn.

2. Cơ sở lý thuyết

2.1. Cơ sở dữ liệu tuần tự

Một chuỗi tuần tự S là một danh sách có thứ tự các phần tử $[i_1, i_2...i_n]$, trong đó $ik \in Z (1 \le k \le n)$ (Gueniche et al., 2015). Một cơ sở dữ liệu tuần tự là 1 tập các chuỗi tuần tự $S = [s_1, s_2...s_n]$ và một tập các phần tử $I = [i_1, i_2, ...i_n]$, trong đó mỗi chuỗi tuần tự $s_x = [X_1, X_2, ... X_n]$ sao cho $X_1, X_2, ...X_n \subseteq I$ (Fournier-Viger, Nkambou, & Tseng, 2011).

2.2. Cây dựng đoán nén (Compact Prediction Tree - CPT)

2.2.1. Khái niệm cây dự đoán nén

Cây dự đoán nén gồm ba cấu trúc phân biệt: (1) Prediction Tree (PT). (2) Lookup Table (LT) và (3) Inverted Index (Gueniche, Fournier-Viger, & Tseng, 2013). Trong suốt quá trình huấn luyện, các chuỗi tuần tự được xem xét để xây dựng dần ba cấu trúc này. Đây là một kiểu của cây tiền tố. Nó chứa tất cả các chuỗi tuần tự huấn luyện. Mỗi nút của cây biểu diễn 1 phần tử và mỗi chuỗi tuần tự huấn luyện được biểu diễn bởi 1 đường dẫn bắt đầu từ gốc của cây và kết thúc bằng một nút trong hay một nút lá.

2.2.2. Phương pháp dự đoán phần tử kế tiếp trên cây dự đoán nén

Theo nghiên cứu (Gueniche et al., 2015), CPT dùng kết quả của mỗi chuỗi tuần tự tương tự với s để thực hiện dự đoán. Đặt $u = j1,j2, ..,jm \langle j_1,j_2,...,j_m \rangle$ là 1 chuỗi tuần tự tương tự với s. Mệnh đề kết quả của u đối với s là chuỗi tuần tự con dài nhất jv.jv+1,...jm $\langle j_v,j_{v+1}, ..j_m \rangle$ của u sao cho $U_{v+1}^{v-1}\{j_k\} \subseteq P_v(s)$ và $1 \le v \le m$. Mỗi phần tử được tìm thấy trong mệnh đề kết quả của 1 chuỗi tuần tự tương tự nhau của s được lưu trong 1 cấu trúc dữ liệu được gọi là *Count Table* (CT). *Count Table* lưu độ hỗ trợ (tần số) của mỗi phần tử này, mà là một ước lượng của $P(e\ P_y(S)|P_x(S))$. CPT trả về phần tử (các phần tử) được hỗ trợ tốt nhất trong CT vì những dự đoán của nó.

3. Giải thuật giảm chiều rộng cây dự đoán nén

Việc giảm chiều rộng cây dự đoán nén được thực hiện bằng cách áp dụng giải thuật (Thon Da & Hanh, 2018)

Dữ liệu nhập vào:

+ arr_sequence: Mảng chứa các chuỗi tuần tự trong cơ sở dữ liệu tuần tự

+ arr_query: Mảng chứa các phần tử trong chuỗi dữ liệu cần dự đoán phần tử kế tiếp

Dữ liệu thu được: Cơ sở dữ liệu tuần tự đã được thu gọn

Chi tiết mã giả (Pseudo Code) của Bước 2 như sau:

Đoạn mã giả trình bày cách loại bỏ các chuỗi không cần thiết ra khỏi cơ sở dữ liệu tuần tự. Đó là các trường hợp các chuỗi trong cơ sở dữ liệu tuần tự chứa 1 chuỗi cần dự đoán duy nhất (dòng lệnh 9 đến dòng lệnh 16). Điều này cũng có nghĩa là các nhánh của cây dự đoán nén có tận cùng là chuỗi cần dự đoán sẽ được loại bỏ ra khỏi cây dự đoán nén ban đầu.

4. Thuật toán nối bảo toàn thông tin

Theo Dr. Yangjun Chen[1] (trường ĐH University of Winnipeg, Canada), thuật toán nối bảo toàn thông tin[2] được mô tả như sau:

Dữ liệu đầu vào: Một quan hệ R, 1 phân rã $D = \{R_1, R_2,..., R_m\}$ của quan hệ R và 1 tập các phụ thuộc hàm F.

1. Tạo một ma trận S với 1 dòng i ứng với mỗi quan hệ Ri trong D, và 1 cột j ứng với mỗi thuộc tính Aj trong R.

2. Đặt $S(i,j) := b_{ij}$ cho tất cả các phần tử trong ma trận.

3. **For** mỗi dòng i biểu diễn quan hệ R_i **Do**
{**For** mỗi cột j biểu diễn Aj **Do**
{**If** quan hệ Ri có chứa Aj **Then**
đặt $S(i, j) := aj$;}

4. Lặp lại vòng lặp sau đến khi không thể tiếp tục được nữa
{**For** mỗi phụ thuộc hàm $X \rightarrow Y$ trong F **Do**
For tất cả các dòng trong S mà có cùng các ký hiệu trong các cột tương ứng với các thuộc tính trong X **Do**

```
1. //Tìm các chuỗi tuần tự có chứa chuỗi cần dự đoán phần tử kế tiếp
2. Cấp phát mảng chuỗi seq có n phần tử
3. k := 0 //k: số lượng các các phần tử trong chuỗi dữ liệu cần dự đoán
4. str_contain_query = " "
5. // str_contain_query là chuỗi chứa chuỗi tuần tự cần dự đoán
6. For i = 0 to (k-1) do
7. If (arr_sequence[i] có chứa ít nhất một phần tử thuộc query) Then
8. Begin
9. If (query ⊆ arr_contain_query[i] and it is not at the last position of ⊆
10. arr_contain_query[i] Or (query ⊆ arr_contain_query[i] and it is at the
11. last position of arr_contain_query[i] And Card{query ⊆
12. arr_contain_query[i]} > 1)) Then
13.     Begin
14.         SD_OK + = arr_contain_query[i] // Chuỗi tuần tự hợp lệ được chọn
15.     End
16. End
```

{

Tạo các kí hiệu trong mỗi cột mà tương ứng với một thuộc tính trong Y là như nhau trong tất cả các dòng như sau:

Nếu bất cứ dòng nào có 1 ký hiệu "a" đối với mỗi cột, đặt những dòng khác cùng ký hiệu "a" trong cột đó.

Nếu không có ký hiệu "a" tồn tại ứng với thuộc tính trong bất kỳ hàng nào, chọn 1 trong các ký hiệu "b" mà xuất hiện trong 1 trong các dòng ứng với thuộc tính đó và đặt các dòng khác cùng ký hiệu "b" trong cột đó.

}

}

5. Nếu một dòng được tạo ra toàn ký hiệu "a", khi đó phân rã là không mất thông tin, ngược lại (không có dòng nào toàn ký hiệu "a" thì phân rã là mất thông tin.

5. Giải pháp kiểm tra bảo toàn thông tin cây dự đoán nén

Xét chuỗi tuần tự Q có k phần tử Q = p1,p2,...,pk ($p_1,p_2,...,p_k$) và cơ sở dữ liệu tuần tự có n phần tử SD = {S_1, S_2, ...S_n}

Xét trường hợp Q tồn tại trong SD, ta nhận thấy rằng, Q có thể có thể xuất hiện trong các vị trí theo từng trường hợp sau trong cơ sở dữ liệu tuần tự SD.

Trường hợp 1: Q đứng trước 1 phần chuỗi tuần tự thuộc 1 chuỗi tuần tự nằm trong SD.

Xét chuỗi tuần tự có dạng QX với X, Q $\subseteq S_i$, trong đó S_i là 1 chuỗi tuần tự trong cơ sở dữ liệu SD.

Xét quan hệ r(Q, X_1, X_2, ..., X_k) và các phân rã của nó:

$R_1(Q, X_1)$ với phụ thuộc hàm $Q \rightarrow X_1$

$R_2(Q, X_2)$ với phụ thuộc hàm $Q \rightarrow X_2$

...

$R_k(Q, X_k)$ với phụ thuộc hàm $Q \rightarrow X_k$

Ta kiểm tra xem phép phân rã trong trường hợp này có có bảo toàn thông tin hay không.

Vì quan hệ gốc có *(k+1)* thuộc tính và được phân rã thành *k* quan hệ, nên ta cần có bảng chứa (k+1) cột và k hàng, ta cũng ánh xạ các thuộc tính thành A_1, A_2, A_3, ...A_k, A_{k+1}

Bước 1:

+ Lược đồ R_1 có các thuộc tính Q và X_1 nên lần lượt các cột $Q(A_1)$ và $X_1(A_2)$, ta ghi nhận a_1 và a_2

+ Lược đồ R_2 có các thuộc tính Q và X_2 nên lần lượt các cột Q (A_1) và X_2 (A_3), ta ghi nhận a_1 và a_3

...

+ Lược đồ R_{k-1} có các thuộc tính Q và X_{k-1} nên lần lượt các cột $Q(A_1)$ và $X_k(A_{k+1})$, ta ghi nhận a_1 và a_{k+1}. Hình ảnh của Bảng 1 như sau:

Bảng 1. Minh họa Bước 1 của Trường hợp 1

	Q (A_1)	X_1 (A_2)	X_2 (A_3)	..	X_{k-1} (A_k)	X_k (A_{k+1})
R_1	a_1	a_2	b_{13}	...	b_{1k}	$b_{1\,(k+1)}$
R_2	a_1	b_{22}	a_3	...	b_{2k}	$b_{2\,(k+1)}$
...	..		...	...	...	. .
R_{k-1}	a_1	$b_{(k-1)\,2}$	$b_{(k-1)\,3}$	..	$b_{(k-1)\,k}$	$b_{(k)\,(k+1)}$
R_k	a_1	b_{k2}	b_{k3}	...	$b_{(k)\,(k)}$	a_{k+1}

Bước 2:

Xét $Q \rightarrow X_1$, ta tìm các giá trị cùng cùng giá trị ở cột Q (A_1). Nhận xét rằng, các giá trị của cột Q (A_1) có cùng giá trị a1, do đó ta sẽ biến đổi để các giá trị ở cột X_1 (A_2) thành toàn a_2

Tương tự, các giá trị của côt Q (A_1) có cùng giá trị a_1, do đó ta sẽ biến đổi để các giá trị ở cột X_2 (A_3) thành toàn a_3. Cuối cùng, Q (A_1) có cùng giá trị a_1, do đó ta sẽ biến đổi để các giá trị ở côt X_k (A_{k+1}) thành toàn a_{k+1}

Bảng 2. Minh họa Bước 2 của Trường hợp 1

	Q (A_1)	X_1 (A_2)	X_2 (A_3)	.	X_{k-1} (A_k)	X_k (A_{k+1})
R_1	a_1	a_2	a_3	...	a_k	a_{k+1}
R_2	a_1	a_2	a_3	...	a_k	a_{k+1}
...	.	a_2	..	...	...	...
R_{k-1}	a_1	a_2	a_3	...	a_k	a_{k+1}
R_k	a_1	a_2	a_3	...	a_k	a_{k+1}

Theo Bảng 2, các phân rã đều có các dòng toàn a nên phép phân rã r(Q, X_1, X_2, ..., X_k) thành các phân rã R_1, R_2..., R_{k-1}, R_k là bảo toàn thông tin.

Hay nói khác hơn, các chuỗi tuần tư trong cơ sở dữ liệu tuần tự có Q đứng trước không bị mất thông tin trong quá trình rút trích các chuỗi tuần tư.

**** Trường hợp 2:*** Q đứng sau một phần chuỗi tuần tự thuộc 1 chuỗi tuần tự nằm trong SD.

Xét chuỗi tuần tư có dạng XQ với X, Q $\subseteq S_i$,: trong đó S_i là 1 chuỗi tuần tự trong cơ sở dữ liệu SD.

Xét quan hệ r(Q, X_1, X_2, ..., X_k) và các phân rã của nó:

$R_1(X_1, Q)$ với phụ thuộc hàm $X_1 \rightarrow Q$

$R_2(X_2, Q)$ với phụ thuộc hàm $X_2 \rightarrow Q$

...

$R_k(X_k, Q)$ với phụ thuộc hàm $X_k \rightarrow Q$

Ta kiểm tra xem phép phân rã trong trường hợp này có có bảo toàn thông tin hay không.

Bước 1: Vì quan hệ gốc có *(k+1)* thuộc tính và được phân rã thành k quan hệ, nên ta cần có bảng chứa *(k+1)* cột và k hàng, ta cũng ánh xạ các thuộc tính thành $A_1, A_2, A_3, .. A_k, A_{k+1}$

Bước 2: Lược đồ R1 có các thuộc tính Q và X_1 nên lần lượt các cột X_1 (A_1) và Q (A_{k+1}), ta ghi nhận a_1 và a_{k+1}.

+ Lược đồ R_2 có các thuộc tính Q và X_2 nên lần lượt các cột X_2 (A_2) và Q (A_{k+1}), ta ghi nhận a_2 và a_{k+1}.

...

Cuối cùng, lược đồ Rk có các thuộc tính Q và X_k nên lần lượt các cột X_k (A_k) và Q (A_{k+1}), ta ghi nhận a_k và a_{k+1}. Hình ảnh của Bảng 3 như sau:

Bảng 3. Minh họa Bước 2 của Trường hợp 2

	X_1 (A_1)	X_2 (A_2)	...	X_{k-1} (A_{k-1})	X_k (A_k)	Q (A_{k+1})
R_1	a_1	b_{12}	.	$b_{1\,(k-1)}$	b_{1k}	a_{k+1}
R_2		a_2				a_{k+1}
...	...		.		.	a_{k+1}
R_{k-1}	$b_{(k-1)\,1}$	$b_{(k-1)\,1}$		a_{k-1}	$b_{(k-1)\,k}$	a_{k+1}
R_k	b_{k1}	b_{k2}	.	$b_{k\,(k-1)}$	a_k	a_{k+1}

Xét $X_1 \rightarrow Q$, ta sẽ tìm các dòng có chung giá trị ở cột X_1 (A_1). Vì không có dòng nào như thế, nên bảng không thay đổi.

Tiếp theo, xét $X_2 \rightarrow Q$, ta sẽ tìm các dòng có chung giá trị ở cột X_2 (A_2). Vì không có dòng nào như thế, nên bảng không thay đổi.

Thực hiện cách tương tự, ta thấy rằng không có dòng nào như thế, nên bảng không thay đổi.

Như vậy, không có dòng nào trong bảng chứa toàn a, nên phép phân rã r r(Q, X_1, X_1, .., X_k) thành $R_1, R_2, ..., R_k$ là ***không bảo toàn thông tin.***

* ***Trường hợp 3:*** Q nằm giữa một phần chuỗi tuần tự thứ nhất (X) và chuỗi tuần tự thứ 2 (Y) thuộc chuỗi tuần tự nằm trong SD. (Bảng 4)

Xét chuỗi tuần tự có dạng XQY với X, Y, $Q \subseteq S_i$, trong đó S_i là 1 chuỗi tuần tự trong cơ sở dữ liệu SD. Xét quan hệ r(X, Q, Y) và các phân rã của nó:

Bảng 4. Minh họa Bước 2 của Trường hợp 3

	X (A_1)	Q (A_2)	Y (A_3)
R_1	a_1	a_2	b_{13}
R_2	b_{12}	a_2	a_3
R_3	a_1	b_{23}	a_3

$R_1(X, Q)$ với phụ thuộc hàm $X \rightarrow Q$

$R_2(Q, Y)$ với phụ thuộc hàm $Q \rightarrow Y$

$R_3(X, Y)$ với phụ thuộc hàm $X \rightarrow Y$

Bước 3: Xét $X \rightarrow Q$, ta tìm các giá trị cùng cùng giá trị ở cột X (A_1). Nhận xét rằng, các giá trị của hàng R_1 và hàng R_3 cột Q (A_1) có cùng giá trị a_1, do đó, ta sẽ biến đổi để các giá trị ở cột Q (A_2) theo các hàng này thành toàn a_2. (Bảng 5)

Bảng 5. Minh họa Bước 3 của Trường hợp 3

	X (A_1)	Q (A_2)	Y (A_3)
R_1	a_1	a_2	b_{13}
R_2	b_{12}	a_2	a_3
R_3	a_1	b_2	a_3

Bước 4: Xét $Q \rightarrow Y$, ta tìm các giá trị cùng cùng giá trị ở cột Q (A_2). Nhận xét rằng các giá trị của cột Q (A_2) có cùng giá trị a_2, do đó, ta sẽ biến đổi để các giá trị ở cột Y(A_3) thành toàn a_3. (Bảng 6)

Bảng 6. Minh họa Bước 4 của Trường hợp 3

	X (A_1)	Q (A_2)	Y (A_3)
R_1	a_1	a_2	a_3
R_2	b_{12}	a_2	a_3
R_3	a_1	a_2	a_3

Ta thấy, các dòng R1 và R3 có chứa toàn R, nên phép phân rã là bảo toàn thông tin.

Vậy, trong 3 trường hợp đã xét, trường hợp tồn tại Q đứng sau 1 phần chuỗi tuần tự thuộc 1 chuỗi tuần tự nằm trong SD là 1 trường hợp mất mát thông tin. Điều này sẽ dẫn đến việc dự đoán không chính xác. Do vậy, việc loại bỏ những chuỗi tuần tự có đặc điểm chỉ có duy nhất Q tận cùng của chuỗi tuần tự là rất cần thiết.

6. Kết quả thực nghiệm và đánh giá

6.1. Tập dữ liệu

Nhóm nghiên cứu tiến hành thực hiện giảm các cây dự đoán nén trên 4 tập dữ liệu được thu thập từ Website của GS. TS Philippe Fournier-Viger, cụ thể như sau:

BMSWebView2 (Gazelle) (KDD CUP 2000) là tập dữ liệu được dùng trong cuộc tranh tài KDD-CUP 2000. Tập dữ liệu này chứa 77,512 chuỗi tuần tự về dữ liệu click-stream. Có 3.340 phần tử khác nhau trong tập BMSWebView2.

KOSARAK là 1 tập dữ liệu click-stream từ một cổng thông tin tin tức của Hungary. Dữ liệu này có gốc từ http://fimi.ua.ac.be/data/. Trong bài báo này, nhóm nghiên cứu sử dụng tập dữ liệu có dạng cơ sở dữ liệu tuần tự gồm có 25,000 chuỗi dữ liệu tuần tự.

LEVIATHAN là 1 tập dữ liệu được chuyển thể từ Tiểu thuyết Leviathan, được viết bởi Thomas Hobbes (sinh năm 1651) sang 1 cơ sở dữ liệu tuần tự (mỗi từ là 1 phần tử). Tập dữ liệu này chứa 5,834 chuỗi tuần tự.

FIFA là 1 tập dữ liệu gồm 20,450 chuỗi tuần tự về dữ liệu click-stream từ website FIFA World Cup 98.

6.2. Các kết quả thực nghiệm

Nhóm nghiên cứu tiến hành giảm chiều rộng cây dự đoán nén theo giải pháp được trình bày ở phần 3 với 5 tập dữ liệu (cơ sở dữ liệu tuần tự) được mô tả ở trên. Kết quả thực nghiệm giải pháp giảm chiều rộng trên cây dự đoán nén được thực hiện trên các 4 tập dữ liệu *BMSWebView2, KOSARAK, LEVIATHAN, FIFA* cho thấy khi phân tích chuỗi dự đoán và loại bỏ các nhánh không cần thiết đã giúp các cây dự đoán được thu gọn hơn. Cụ thể, với cây dự đoán nén trên tập dữ liệu *BMSWebView2*, trong trường hợp chuỗi dự đoán ⟨314917,314921⟩ khi áp dụng giải pháp giảm chiều rộng đã giảm đến 77.497 nhánh. Với cây dự đoán nén trên tập dữ liệu *KOSARAK*, trong trường hợp chuỗi dự đoán ⟨3, 103, 64⟩ khi áp dụng giải pháp giảm chiều rộng đã giảm 69.975 nhánh. Với cây dự đoán nén trên tập dữ liệu *LEVIATHAN*, trong trường hợp chuỗi dự đoán ⟨125, 299, 30⟩ khi áp dụng giải pháp giảm chiều rộng đã giảm 5.814 nhánh. Với cây dự đoán nén trên tập dữ liệu *FIFA*, trong trường hợp chuỗi dự đoán ⟨466,80,28⟩ khi áp dụng giải pháp giảm chiều rộng đã giảm 20.402 nhánh.

7. Kết luận

Bài báo này đã đề xuất một giải pháp cho việc giảm chiều rộng cây dự đoán nén dựa vào phân tích chuỗi dự đoán đầu vào. Khi cây dự đoán nén được loại bỏ các nhánh không cần thiết, thời gian truy xuất dữ liệu, tìm kiếm trên cây cũng như việc dự đoán sẽ nhanh hơn rất nhiều lần so với ban đầu. Các cây dự đoán nén đã thu hẹp số nhánh rất nhiều khi triển khai thử nghiệm giải pháp giảm chiều rộng trên các cơ sở dữ liệu tuần tự BMSWeb-View2, KOSARAK, LEVIATHAN, FIFA trên các chuỗi dự đoán khác nhau và các cây dự đoán nén thu được đều có chiều rộng rất nhỏ so với các cây dự đoán nén ban đầu (khi chưa áp dụng giải pháp giảm chiều rộng).

Tuy nhiên, việc giảm chiều rộng trên cây dự đoán nén tùy thuộc vào mật độ xuất hiện của chuỗi dự đoán ở phần cuối các nhánh mỗi cây dự đoán nén. Nếu cây dự đoán nén có càng nhiều nhánh chứa chuỗi dự đoán ở tận cùng các nhánh, sẽ giảm được càng nhiều nhánh.

Nhóm nghiên cứu cũng đưa ra các lập luận để chứng minh giải pháp giảm chiều rộng cho cây dự đoán nén bằng cách loại bỏ các nhánh mà tận cùng là chuỗi cần dự đoán bằng thuật toán nối bảo toàn ■

Lời cám ơn:

Nghiên cứu được tài trợ bởi Đại học Quốc gia Thành phố Hồ Chí Minh (ĐHQG-HCM) trong khuôn khổ Đề tài mã số C2019-34-06.

TÀI LIỆU TRÍCH DẪN:

[1]*http://ion.uwinnipeg.ca/~ychen2/*

[2]*http://ion.uwinnipeg.ca/~ychen2/databaseNotes/lossless-jam.ppt*

TÀI LIỆU THAM KHẢO:

1. Bernard, G., & Andritsos, P (2019). Accurate and Transparent Path Prediction Using Process Mining. Paper presented at the European Conference on Advances in Databases and Information Systems.

2. Fournier-Viger, P., Nkambou, R., & Tseng, V. S.-M. (2011). RuleGrowth: Mining sequential rules common to several sequences by pattern-growth. Paper presented at the Proceedings of the 2011 ACM symposium on applied computing.

3. Gueniche, T., Fournier-Viger, P., Raman, R., & Tseng, V. S. (2015). CPT+: Decreasing the time/space complexity of the Compact Prediction Tree. Paper presented at the Pacific-Asia Conference on Knowledge Discovery and Data Mining.

4. Gueniche, T., Fournier-Viger, P., & Tseng, V. S. (2013). Compact prediction tree: A lossless model for accurate sequence prediction. Paper presented at the International Conference on Advanced Data Mining and Applications.

5. Previde, P. (2019). Applications of data mining to student performance prediction and curriculum design. San Francisco State University.

6. Rjeily, C. B., Badr, G., El Hassani, A. H., & Andres, E. (2017). Sequence prediction algorithm for heart failure prediction. Paper presented at the International Conference e-Health.

7. Sun, R., & Giles, C. L. (2001). Sequence learning: from recognition and prediction to sequential decision making. IEEE Intelligent Systems, 16(4), 67-70.

8. Thon Da, N., & Hanh, T. (2018). A novel approach based on sequence prediction for webpage access. 2018, 7(4), 4. doi:10.14419/ijet.v7i4.13901

Ngày nhận bài: 8/2/2020
Ngày phản biện đánh giá và sửa chữa: 18/2/2020
Ngày chấp nhận đăng bài: 28/2/2020

Thông tin tác giả:
NGUYỄN THÔN DÃ
Khoa Hệ thống thông tin, Trường Đại học Kinh tế - Luật

THE SOLUTION FOR REDUCING THE WIDTH OF COMPACT PREDICTION TREE WITHOUT LOSING INFORMATION

● **NGUYEN THON DA**
Faculty of Management Informatics
University of Economics and Law,
Vietnam National University - Ho Chi Minh City

ABSTRACT:

This paper proposes a solution that reduces the width of compact prediction tree (CPT) without losing information. The strategy of this solution is to rely on predictive sequence analysis to reduce the width of the CPT and check the information conservation by using the lossless-join algorithm. The application of CPT or CPT+ models into sequential sequences prediction is helpful because the reduction in size will optimize the retrieval, searching and prediction and also cut the time and memory cost.

Keywords: CPT, CPT+, information security, compact prediction tree.