**Đề tài: Tìm hiểu các thuật toán khai phá tập hữu ích cao và các bài toán ứng dụng**

* **GNN**

PHẦN MỞ ĐẦU

1.1 Tính cấp thiết của đề tài

1.2. Mục đích của đề tài

1.3. Cách tiếp cận và phương pháp nghiên cứu

1.3.1. Đối tượng nghiên cứu

1.3.2. Phạm vi nghiên cứu

1.4. Phương pháp nghiên cứu

1.5. Kết quả dự kiến đạt được

PHẦN NỘI DUNG

Chương 1: Cơ Sở Lý Thuyết

1.2. Tập hữu ích cao và bài toán khai phá độ hữu ích cao

1.2.1. Tập phổ biến

1.2.2. Tập hữu ích cao

1.2.2.1. Khái niệm tập hữu ích cao

1.2.2.2. Định nghĩa bài toán

1.2.2.3. Ứng dụng khai thác tập hữu ích cao

1.2.3. Một số thuật toán khai phá tập mục độ hữu ích cao

1.2.3.1. Level-wise

1.2.3.2. Tree-based

1.2.3.3. Utility-list based

1.2. Hệ thống gợi ý (Recommender Systems - RS)

1.2.1 Giới thiệu

1.2.2 Lợi ích

1.2.3 Các kỹ thuật chính trong hệ thống gợi ý

1.2.3.1 Content based filtering

1.2.3.2 Collaborative filtering (CF) Techniques

1.2.3.3 Hybrid recommendation.

1.3. Tổng quan về mạng nơ-ron học sâu

1.3.1 Mạng Nơ-ron nhân tạo

1.3.2. Mạng học sâu

1.3.3. Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN)

1.3.3.1. Graph hay đồ thị là gì?

1.3.3.2. Giới thiệu về GNN

1.3.3.3. Các thành phần chính của GNN

1.3.3.4. Cách thức hoạt động của GNN

Chương 2: Thiết Kế Hệ Thống

2.1.

2.2.

Chương 3: Phát Triển và Triển Khai

3.1.

3.2.

Chương 4: Thực Nghiệm và Đánh Giá

4.1 Kết quả đạt được của đề tài

4.2 Hạn chế

4.3 Hướng phát triển

**Phần KẾT LUẬN**

Tài liệu tham khảo

[1]…

[2]…

**KẾ HOẠCH THỰC HIỆN**

(Ghi rõ từng thời gian tiến hành các công việc của luận văn, thời gian thí nghiệm, thăm quan hiện trường, ....)

| **STT** | **Thời gian** | **Công việc** | **Ghi chú** |
| --- | --- | --- | --- |
| **1** |  |  |  |
| **2** |  |  |  |
| **3** |  |  |  |
| **4…** |  |  |  |

**Đề tài: Tìm hiểu các thuật toán khai phá tập hữu ích cao và các bài toán ứng dụng**

* **GNN**

**PHẦN MỞ ĐẦU**

1.1 Tính cấp thiết của đề tài

1.2. Mục đích của đề tài

1.3. Cách tiếp cận và phương pháp nghiên cứu

1.3.1. Đối tượng nghiên cứu

1.3.2. Phạm vi nghiên cứu

1.4. Phương pháp nghiên cứu

1.5. Kết quả dự kiến đạt được

**PHẦN NỘI DUNG**

## Chương 1: Cơ Sở Lý Thuyết

### 1.2. Tập hữu ích cao và bài toán khai phá độ hữu ích cao

#### 1.2.1. Tập phổ biến

Cho tập I={i1, i2, i3, … , in} là tập các sự kiện khác nhau. Cơ sở dữ liệu giao dịch DB={t1,t2,t3, …,tm} là tập hợp các giao dịch ti với 𝑖 ∈ [0, 𝑚], và 𝑡𝑖 ⊂ 𝐼. Mẫu 𝑃 ⊂ 𝐼 được gọi là xuất hiện bên trong giao dịch T nếu và chỉ nếu mọi phần tử của P đều thuộc tập T. Ví dụ: P={a,b,c} và T={a,b,c,d} khi đó P được gọi là xuất hiện trong tập T. Độ hỗ trợ của mẫu P là số lần xuất hiện trong cơ sở dữ liệu giao dịch DB, mỗi giao dịch Ti có chứa P được tính là một lần. Độ dài của mẫu P được tính bằng số item có trong P. Một mẫu có độ dài k còn được gọi là tập k-itemsets. Độ hỗ trợ của Pđược ký hiệu là 𝑠𝑢𝑝(𝑃) .Độ hỗ trợ tối tiểu 𝜑 là một ngưỡng do người dùng đặt ra.

Bài toán khai thác mẫu phổ biến được phát biểu như sau: Cho cơ sở dữ liệu giao dịch DB và một ngưỡng hỗ trợ tối tiểu 𝜑 yêu cầu tìm ra tất cả các mẫu 𝑃 ⊂ 𝐼 sau cho sup(𝑃) ≥ 𝜑

#### 1.2.2. Tập hữu ích cao

#### 1.2.2.1. Khái niệm tập hữu ích cao

#### 1.2.2.2. Định nghĩa bài toán

| Tid | Các giao dịch |
| --- | --- |
| T1 | (A,1) (C,1) (D,1) |
| T2 | (A,2) (C,6) (E,2) (G,5) |
| T3 | (A,1) (B,2) (C,1) (D,6) (E,1) (F,5) |
| T4 | (B,4) (C,3) (D,3) (E,1) |
| T5 | (B,2) (C,2) (E,1) (G,2) |

Bảng biển diễn giá trị lợi nhuận của các mục:

| Danh mục | A | B | C | D | E | F | G |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lợi nhuận | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

**Định nghĩa 1:** Độ hữu ích (TU) của một giao dịch Tc là tổng độ hữu ích của các các item trong giao dịch Tc, tức là TU(Tc) =

**Ví dụ:** TU(T1) = 1\*5+1\*1+1\*2 = 8

**Định nghĩa 2:** Lợi ích trọng số giao dịch (TWU) của 1 itemset X được định nghĩa bằng tổng lợi ích giao tác của các giao dịch chứa X, tức là:

TWU(X) =

**Ví dụ:** TWU(A) = TU(T1) +TU(T2)+TU(T3) = 8+27+30 = 65

**Định nghĩa 3:** Giá trị tiện ích tối thiểu được chỉ định là giá trị tiện ích tối thiểu tuyệt đối, được ký hiệu là minutil và được tính như sau:

Minutil =

**Ví dụ:** Giả sử , khi đó minutil được tính như sau:

Minutil = \* (TU(T1) + TU(T2) + TU(T3) + TU(T4) + TU(T5)

= 28% \* (8 + 27 + 30 + 20 + 11) = 0.28 \* 96 = 26,88

**Định nghĩa 4:** Độ hữu ích còn lại của 1 itemset X còn lại trong giao dịch Tc(X ) được ký hiệu là RU(X,Tc) hay còn được viết là Rutil được tính là:

RU(X,Tc) =

**Ví dụ:** RU(C,T1) = u(A)+u(D) = 5+2 = 7

**Định nghĩa 5:** Độ hữu ích của 1 itemset X, ký hiệu u(X) là tổng tiện ích của X trong tất cả các giao dịch có chứa X trong D trong đó u(X) sẽ được tính như sau:

u(X) =

**Ví dụ:** u(A) = u(A,T1) + u(A,T2) + u(A,T3) = 5+10+5 = 20

**Định nghĩa 6:** Một giao dịch được coi là đã được sửa đổi là sau khi tất cả các mục có tiện ích có trọng số giao dịch nhỏ hơn một minutil đã cho sẽ bị xóa khỏi giao dịch; các mục còn lại được sắp xếp theo thứ tự tăng dần theo trọng số giao dịch.

**Ví dụ:** Giả sử cho minutil = 60 và giá trị lợi ích trọng số giao dịch (TWU) lần lượt là:

| Danh mục | A | B | C | D | E | F | G |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TWU | 65 | 61 | 96 | 58 | 88 | 30 | 38 |

Với giá trị minutil >= 60 từ các danh mục bị loại ra khỏi danh sách là D, F, G. Ta tiến hành sắp xếp lại các danh mục theo thứ tự từ nhỏ đến lớn theo giá trị TWU và sẽ được là: **B < A < E < C**

Khi đó bảng lợi ích sẽ được sửa đổi như sau:

| Tid | Item Util | Item Util | Item Util | Item Util | TU |
| --- | --- | --- | --- | --- | --- |
| T1 | A 5 | C 1 |  |  | 6 |
| T2 | A 10 | C 6 |  |  | 16 |
| T3 | B 4 | A 5 | E 3 | C 1 | 13 |
| T4 | B 8 | E 3 | C 3 |  | 14 |
| T5 | B 4 | E 3 | C 2 |  | 9 |

**Định nghĩa 7:** Cho một mục X và một giao dịch (hoặc mục) T với X⊆T, tập hợp tất cả các mục sau X trong T được ký hiệu là T / X.

**Ví dụ:** T1/{C} = {AD}

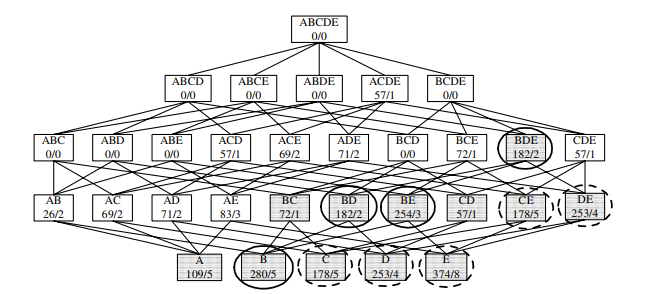
#### 1.2.2.3. Ứng dụng khai thác tập hữu ích cao

#### 1.2.3. Một số thuật toán khai phá tập mục độ hữu ích cao

Các thuật toán khai thác vật phẩm tiện ích cao đã được đề xuất như UMining, Two-Phase, IHUP , UP-Growth , HUP-Growth , MUGrowth, HUI-Miner , FHM, ULB-Miner , HUI-Miner và EFIM . Tất cả các thuật toán này có cùng đầu vào và cùng một đầu ra. Sự khác biệt giữa các thuật toán này nằm ở cấu trúc dữ liệu và chiến lược được sử dụng để tìm kiếm các mục tiện ích cao. Cụ thể hơn, các thuật toán khác nhau ở (1) liệu chúng sử dụng tìm kiếm đầu tiên theo chiều sâu hay chiều rộng, (2) loại biểu diễn cơ sở dữ liệu mà chúng sử dụng bên trong hay bên ngoài, (3) cách chúng tạo hoặc xác định các bộ mục tiếp theo sẽ được khám phá trong không gian tìm kiếm và (4) cách chúng tính toán tiện ích của các bộ mục để xác định xem chúng có đáp ứng ràng buộc tiện ích tối thiểu hay không. Các lựa chọn thiết kế này ảnh hưởng đến hiệu suất của các thuật toán này về thời gian thực thi, sử dụng bộ nhớ và khả năng mở rộng, cũng như mức độ dễ dàng thực hiện và mở rộng các thuật toán này cho các tác vụ khai thác dữ liệu khác. Nói chung, tất cả các thuật toán khai thác vật phẩm tiện ích cao đều được lấy cảm hứng từ các thuật toán khai thác vật phẩm thường xuyên cổ điển, mặc dù chúng cũng đưa ra những ý tưởng mới để đối phó với thực tế là biện pháp tiện ích không đơn điệu cũng không đơn điệu.

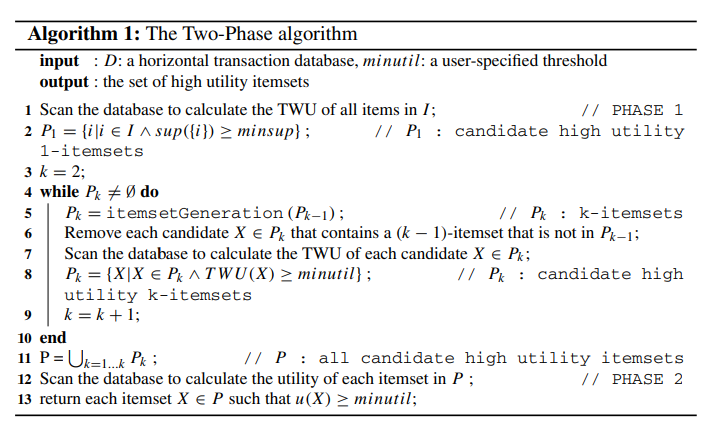
#### 1.2.3.1. Level-wise

##### Two-PhaseThuật toán Two-Phase - Hai pha khái quát hóa thuật toán Apriori, được đề xuất để khai thác các vật phẩm thường xuyên. Hai giai đoạn khám phá không gian tìm kiếm của các mục bằng cách sử dụng tìm kiếm đầu tiên theo chiều rộng. Thuật toán tìm kiếm đầu tiên theo chiều rộng trước tiên xem xét các mục đơn lẻ (1-itemsets), Trong ví dụ đang chạy, đó là {a}, {b}, {c}, {d} và {e}. Sau đó, Two-Phase tạo ra 2-itemset như {a, b},{a, c},{a, d}, và sau đó là 3-itemsets, v.v., cho đến khi nó tạo ra các itemset lớn nhất {a, b, c, d, e} chứa tất cả các mục.



ĐL1: (Cắt tỉa không gian tìm kiếm bằng TWU) Đối với bất kỳ mục X nào, nếu TWU (X) < minutil, thì X là một vật phẩm tiện ích thấp cũng như tất cả các siêu tập hợp của nó.

Mã giả của thuật toán :



**Đầu vào** : cơ sở dữ liệu giao dịch định lượng và ngưỡng minutil làm đầu vào.

**Đầu ra**: Danh sách tập hữu ích cao

**PHA 1**

**1**: Two-Phase quét cơ sở dữ liệu để tính toán TWU của mỗi 1-itemset

**2:** Xác định tập hợp tất cả các ứng viên mục tiện ích cao (**TWU (X) ≥ minuti**l)

**4 -10**: Sau đó, Two-Phase thực hiện tìm kiếm đầu tiên theo chiều rộng để tìm các bộ mục tiện ích cao ứng cử viên lớn hơn (**dòng 4–10**).

Trong quá trình tìm kiếm, Hai pha sử dụng các bộ vật phẩm tiện ích cao ứng cử viên có độ dài nhất định k − 1 (ký hiệu là Pk−1) để tạo ra các mục có độ dài k (ký hiệu là Pk). Điều này được thực hiện bằng cách kết hợp các cặp vật phẩm tiện ích cao ứng cử viên có độ dài k chia sẻ tất cả trừ một mục **(dòng 5).**

Ví dụ: nếu các bộ 1 mục tiện ích cao ứng cử viên là {a},{b},{c} và {e}, Hai pha kết hợp các cặp của các bộ mục này để có được các bộ 2 mục sau: {a, b}, {a, c}, {a, e}, {b, c}, {b, e} và {c, e}. Sau khi tạo các bộ mục có độ dài k, Hai pha sẽ kiểm tra xem các tập con (k - 1) của mỗi mục có phải là các bộ mục tiện ích cao hay không. Nếu một itemset X có một tập con (k − 1) không phải là một ứng cử viên high utility itemset, X không thể là một item tiện ích cao (nó sẽ vi phạm ĐL1) và do đó nó sẽ bị xóa khỏi tập hợp các k-itemset.

VD: tập a có twu < miniutil thì loại tập ab, ac, ad, ad,...

Sau đó, Two-Phase quét cơ sở dữ liệu để tính toán TWU của tất cả các mục còn lại trong Pk **(dòng 7).** Mỗi item có TWU không nhỏ hơn minutil được thêm vào tập hợp Pk của các k-item tiện ích cao (**dòng 8)**. Quá trình này là lặp đi lặp lại cho đến khi không có ứng cử viên nào có thể tạo ra các vật phẩm tiện ích cao.

**PHA 2:**

**12-13**:. Two-Phase quét cơ sở dữ liệu để tính toán chính xác tiện ích của từng bộ mục tiện ích cao của từng ứng viên. Tập hợp tất cả các bộ vật phẩm tiện ích cao ứng cử viên có tiện ích lớn hơn minutil là các bộ vật phẩm tiện ích cao (u(x) > miniutil). Chúng được trả lại cho người dùng **(dòng 13)**

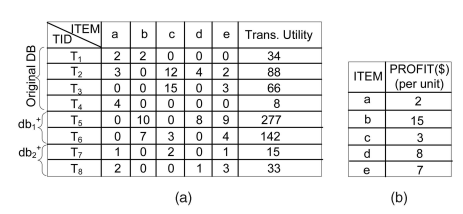
Còn 1 số thuật toán khác như: UMining & UMining\_H (2006),...

#### 1.2.3.2. Tree-based

##### IHUP (2009)

Thuật toán IHUP (Incremental High Utility Pattern) là một phương pháp khai thác các tập hợp mục có lợi ích cao trong các cơ sở dữ liệu cập nhật dần. Mục tiêu của IHUP là khai thác hiệu quả các tập hợp mục hữu ích trong các cơ sở dữ liệu mà dữ liệu mới có thể được thêm vào hoặc dữ liệu cũ có thể bị xóa đi.

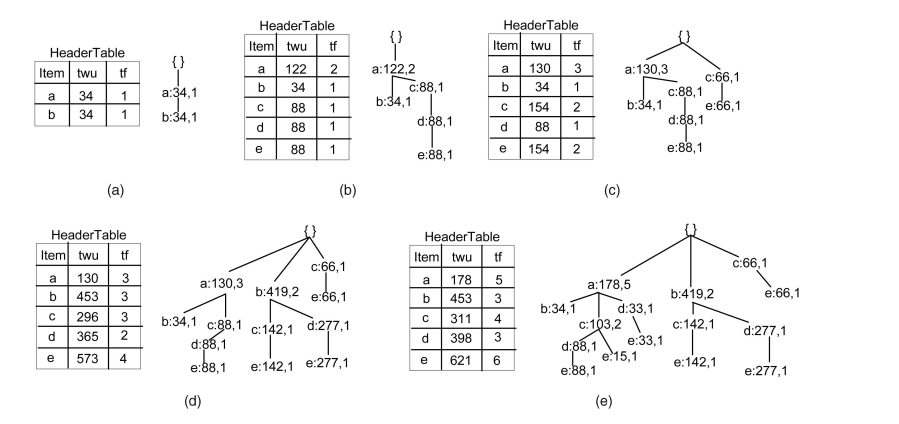
Có ba cấu trúc cây mới để khai thác mô hình tiện ích cao trong cơ sở dữ liệu gia tăng: IHUPL (IHUP with Lexicographic Tree), IHUPTF (IHUP Tree with Frequency), IHUPTWU (IHUP Tree with Transaction Weighted Utilization).



Hình 3:

**ĐN8:** Tần suất giao dịch (tf) của một ip mục là tf (ip) và đại diện cho số lượng giao dịch mà mục đó xuất hiện. Tần số ban đầu của ip là f(ip), biểu thị số lần xuất hiện thực tế của IP trong các giao dịch đó. Ví dụ: tf(c) = 4 như xuất hiện trong T2, T3, T6; và T7 và f(c) = 12 + 15 + 3 + 2 = 32

**IHUPL (IHUP with Lexicographic Tree)**



Hình 4: Cấu tạo của IHUPL-Tree (a) sau khi chèn T1, (b) sau khi chèn T2 (c) sau khi chèn T3 và T4, (d) sau khi chèn db1+ (T5 và T6), và (e) sau khi chèn db2+ (T7 và T8).

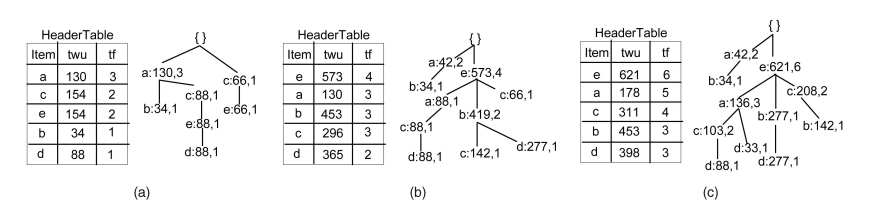
Xem xét cơ sở dữ liệu giao dịch và bảng tiện ích trong Hình 3. Ngay từ đầu, cơ sở dữ liệu chỉ chứa bốn giao dịch. Sau đó, cơ sở dữ liệu ban đầu được tăng lên bằng cách thêm hai nhóm giao dịch. Quá trình xây dựng cơ sở dữ liệu ban đầu (T1-T4) của IHUPL-Tree được mô tả trong Hình 4a, 4b và 4c. Giá trị tu của T1 là 34 trong Hình 3. Hình 4a cho thấy khi T1 được thêm vào cây, nút đầu tiên theo thứ tự từ điển là "a" với twu = 34 và tf = 1. Nút thứ hai là "b" với cùng giá trị. Hình 4b cho thấy khi T2 được thêm vào cây. Tu của T2 là 88, vì vậy nhánh theo thứ tự từ điển là "a: 88, 1", "c: 88, 1", "d: 88, 1" và "e: 88, 1." Ở đây "a" được gán chia sẻ tiền tố với nút "a" hiện có, giá trị twu của nút "a" là 34 + 88 = 122 và giá trị tf là 1 + 1 = 2. Đối với các mục khác, các nút mới được tạo, như thể hiện trong Hình 4b. Sau đó, T3 và T4 được thêm vào để hoàn thành cơ sở dữ liệu ban đầu (Hình 4c). Bằng cách làm theo quy trình tương tự, db1+ (T5 và T6) và db2+ (T7 và T8) được chèn vào IHUPL-Tree, như thể hiện trong Hình 2d và 2e.

Có thể dễ dàng thực hiện các thao tác xóa và sửa đổi trong IHUPL-Tree. Giả sử rằng chúng ta muốn xóa T7, thì giá trị tu của T7 và một giá trị tf có thể dễ dàng giảm từ đường dẫn "ace" trong cây được hiển thị trong Hình 4e. Sau đó, khi giá trị twu và tf của nút "e" trong đường dẫn đó trở thành số không, chúng ta phải xóa nút "e" khỏi nhánh đó. Vì vậy, các giá trị twu và tf được sửa đổi của "a" sẽ lần lượt là 163 và 4 trong nhánh đó. Theo cách tương tự, các giá trị twu và tf đã sửa đổi của "c" sẽ lần lượt là 88 và 1 trong nhánh đó. Giả sử rằng chúng ta không muốn xóa T7 mà chỉ cần sửa đổi nó. Chúng ta muốn giảm số lượng mặt hàng "c" từ hai xuống một. Để thực hiện sửa đổi này, giá trị twu của T7 sẽ giảm 1 x 3 = 3. Chúng ta phải giảm số tiền ba (giá trị twu) từ nhánh đó. Vì IHUPL-Tree luôn duy trì thứ tự từ điển trong cả bảng tiêu đề và nút cây, thuộc tính sau luôn đúng với nó:

Đặc tính 1. Thứ tự các mục trong IHUPL-Tree không bị ảnh hưởng mặc dù thay đổi tần suất của các mục bằng cách thêm, xóa và sửa đổi.

**IHUPTF (IHUP Tree with Frequency)**

Để giảm kích thước của IHUPL-Tree, phải tăng chia sẻ tiền tố bên trong nó. Trong cây này, các nút được sắp xếp theo thứ tự giảm dần theo tần suất giao dịch (tf) của chúng để các mục xảy ra trong nhiều giao dịch có thể được giữ ở phần trên của cây, và do đó, có thể đạt được chia sẻ tiền tố cao hơn. IHUPTF-Tree có thể được xây dựng từ IHUPL-Tree bằng phương pháp điều chỉnh đường dẫn dựa trên kỹ thuật phân loại bong bóng bất cứ lúc nào. Bất kỳ nút nào cũng có thể được chia nhỏ khi nó cần được hoán đổi với bất kỳ nút con nào có số lượng nhỏ hơn nút đó. Nếu số lượng hỗ trợ của cả hai nút bằng nhau, một thao tác trao đổi đơn giản giữa chúng được thực hiện. Sau khi thực hiện mỗi thao tác, các nút hoán đổi có cùng mục sẽ được hợp nhất.



**IHUPTWU (IHUP Tree with Transaction Weighted Utilization)**

Cấu trúc cây: Sử dụng một cây IHUP chứa thông tin về TWU của các mục trong các giao dịch.

Cập nhật: Khi có giao dịch mới được thêm vào hoặc giao dịch cũ bị xóa, cây IHUPTWU sẽ được cập nhật tương ứng với TWU của các mục.

…………………………………………………Lười tìm hiểu quá@@

##### UP-Growth (2010) & UP\_Growth+ (2013)

##### 

Ngoài ra còn nhiều thuật toán khác: MU-Growth (2014), EFIM-Closed (2016), …

#### 1.2.3.3. Utility-list based

##### HUI Miner

**GIỚI THIỆU THUẬT TOÁN**

Để xác định tập lợi ích cao, các thuật toán đầu tiên tạo ra tập ứng cử viên từ cách đánh giá các lợi ích cao vì vậy sẽ gặp những vấn đề là tạo ra một số lượng lớn tập ứng viên nhưng hầu hết các ứng cử viên được sinh ra là lợi ích không cao sau khi các lợi ích được tính chính xác. HUI- Miner (High Utility Itemset Miner) 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 của HUI- Miner. Bằng cách tránh tạo ra các tập ứng viên thế hệ và tính toán lợi ích của nhiều tập ứng viên, HUI- Miner hiệu quả hơn vì có thể khai thác tập lợi ích cao từ danh sách lợi ích (utility list)

**THUẬT TOÁN HUI- MINER**

| Tid | Các giao dịch |
| --- | --- |
| T1 | (A,1) (C,1) (D,1) |
| T2 | (A,2) (C,6) (E,2) (G,5) |
| T3 | (A,1) (B,2) (C,1) (D,6) (E,1) (F,5) |
| T4 | (B,4) (C,3) (D,3) (E,1) |
| T5 | (B,2) (C,2) (E,1) (G,2) |

Bảng biển diễn giá trị lợi nhuận của các mục:

| Danh mục | A | B | C | D | E | F | G |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lợi nhuận | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

Bảng lợi ích:

| **Tid** | **A** | **B** | **C** | **D** | **E** | **F** | **G** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **T1** | 5 | 0 | 1 | 2 | 0 | 0 | 0 |
| **T2** | 10 | 0 | 6 | 0 | 6 | 0 | 5 |
| **T3** | 5 | 4 | 1 | 12 | 3 | 5 | 0 |
| **T4** | 0 | 8 | 3 | 6 | 3 | 0 | 0 |
| **T5** | 0 | 4 | 2 | 0 | 3 | 0 | 2 |

Tính giá trị lợi ích giao tác (TU) của các giao tác T1, T2, T3, T4, T5 khi thực thi là:

T1 = 5+1+2 = 8, T2 = 10+6+6+5 = 27, T3 = 5+4+1+12+3+5 = 30

T4 = 8+3+6+3 = 20, T5 = 4+2+3+2 = 11

| TID | T1 | T2 | T3 | T4 | T5 |
| --- | --- | --- | --- | --- | --- |
| TU | 8 | 27 | 30 | 20 | 11 |

Tính lợi ích trọng số (TWU) của các danh mục khi thực thi:

A = T1+T2+T3 = 8+27+30 = 65

B = T3+T4+T5 = 30+20+11 = 61

C = T1+T2+T3+T4+T5 = 8+27+30+20+11 = 96

D = T1+T3+T4 = 8+30+20 = 58

E = T2+T3+T4+T5 = 27+30+20+11 = 88

F = T3 = 30

G = T2+T5 = 27+11 = 38

| Danh mục | A | B | C | D | E | F | G |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TWU | 65 | 61 | 96 | 58 | 88 | 30 | 38 |

Trong thuật toán HUI-Miner mỗi mục chứa một danh sách tiện ích. Danh sách tiện ích ban đầu lưu trữ thông tin tiện ích về cơ sở dữ liệu được khai thác. Các tiện ích có trọng số giao dịch của tất cả các mục được tích lũy bằng cách quét cơ sở dữ liệu. Nếu tiện ích có trọng số giao dịch của một danh mục nhỏ hơn một minutil nhất định, thì mặt hàng đó không còn được xem xét trong quy trình khai thác tiếp theo. Đối với các mục có tiện ích có trọng số giao dịch vượt quá minutil, chúng được sắp xếp theo thứ tự tăng dần theo giao dịch có trọng số giao dịch. Đối với cơ sở dữ liệu trên, giả sử minutil là 40, và sau đó thuật toán không còn xem xét các mục **F** và **G** sau khi quét cơ sở dữ liệu đầu tiên. Các mục còn lại được sắp xếp: **D < B < A < E < C**

| Tid | Item Util | Item Util | Item Util | Item Util | Item Util | TU |
| --- | --- | --- | --- | --- | --- | --- |
| T1 | D 2 | A 5 | C 1 |  |  | 8 |
| T2 | A 10 | E 6 | C 6 |  |  | 22 |
| T3 | D 12 | B 4 | A 5 | E 3 | C 1 | 25 |
| T4 | D 6 | B 8 | E 3 | C 3 |  | 20 |
| T5 | B 4 | E 3 | C 2 |  |  | 9 |

Mỗi phần tử trong danh sách tiện ích của các mục X chứa ba trường: tid, iutil và rutil.

• Trường tid chỉ ra một giao dịch T có chứa X.

• Trường iutil là tiện ích của X trong T, tức là, u (X, T).

• Trường rutil là tiện ích còn lại của X trong T, tức là ru (X, T).

{D}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 2 | 6 |
| T3 | 12 | 13 |
| T4 | 6 | 14 |
| Tổng | 20 | 33 |

{B}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 4 | 9 |
| T4 | 8 | 6 |
| T5 | 4 | 5 |
| Tổng | 16 | 20 |

{A}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 5 | 1 |
| T2 | 10 | 12 |
| T3 | 5 | 4 |
| Tổng | 20 | 17 |

{E}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T2 | 6 | 6 |
| T3 | 3 | 1 |
| T4 | 3 | 3 |
| T5 | 3 | 2 |
| Tổng | 15 | 12 |

{C}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 1 | 0 |
| T2 | 6 | 0 |
| T3 | 1 | 0 |
| T4 | 3 | 0 |
| T5 | 2 | 0 |
| Tổng | 13 | 0 |

Các mục trên được tính như sau:

Ví dụ danh mục {D}:

u(D,T1) = 2, ru(D,T1) = u(A,T1) + u(C,T1) = 5+1 = 6

Những phần còn lại được xử lý theo các tương tự, đối với những danh mục ở sau việc tính ru(X,T) sẽ được lấy tổng những phần tử sau nó theo thứ tự từ trước là:

**D < B < A < E < C.**

Sau khi xét danh sách tiện ích của 1 phần tử ta đến xét tiếp danh sách của 2 phần tử để dễ nhìn và hình dung nên nhóm sẽ đưa ra một sơ đồ như sau:

Khi đã xác định được danh sách tiện ích của 1 mục thì ta tiến hành xet danh sách tiện ích của 2 mục. Về tổng quát thì danh sách tiện ích của 2 mục {xy} có thể được xây dựng bằng giao điểm của danh sách tiện ích của {x} và của {y} sau đó ta sẽ đi tính các giá trị Iutil và Rutil của {xy}, các giá trị này sẽ được tính như là cách tính của 1 danh mục nhưng ở đây chúng ta sẽ lấy tổng của {x} và {y}.

Lần lượt hội item d với các phần tử đứng phía sau d trong quan hệ thứ tự **D < B < A < E < C** để tạo ra tập item mới: { D,B }, { D,A },{ D,E },{ D,C }, tính UL cho các tập mới phát sinh này.

{DB}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 16 | 9 |
| T4 | 14 | 6 |
| Tổng | 30 | 15 |

{DA}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 7 | 1 |
| T3 | 17 | 4 |
| Tổng | 24 | 5 |

{DE}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 15 | 1 |
| T4 | 9 | 3 |
| Tổng | 24 | 4 |

{DC}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 3 | 0 |
| T3 | 13 | 0 |
| T4 | 9 | 0 |
| Tổng | 33 | 0 |

Tương tự ta sẽ đi tính toán những danh mục còn lại:

{BA}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 9 | 4 |
| Tổng | 9 | 4 |

{BE}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T4 | 11 | 3 |
| Tổng | 11 | 3 |

{BC}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T4 | 11 | 0 |
| T5 | 6 | 0 |
| Tổng | 17 | 0 |

{AE}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T2 | 16 | 6 |
| T3 | 8 | 1 |
| Tổng | 24 | 7 |

{AC}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T1 | 6 | 0 |
| T2 | 16 | 0 |
| T3 | 6 | 0 |
| Tổng | 28 | 0 |

{EC}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T2 | 12 | 0 |
| T3 | 4 | 0 |
| T4 | 6 | 0 |
| T5 | 5 | 0 |
| Tổng | 27 | 0 |

{DBA}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 21 | 4 |
| Tổng | 21 | 4 |

{DBE}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 19 | 1 |
| T4 | 17 | 3 |
| Tổng | 36 | 4 |

{DBC}

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 17 | 0 |
| T4 | 17 | 0 |
| Tổng | 34 | 0 |

Lần lượt tính các giá trị tổng của iulti và rulti của tập item có 1 phần tử, 2 phần tử và k phần tử (k≥3) để tìm ra các tập tiện ích cao với giá trị minutil >= 40

Đầu tiên ta xét danh mục D: có giá trị tổng iutils = 20, tổng giá trị Rutils = 33, ta có iutils + rutils = 20 + 33 = 53 > 40 nên:

Gán tập exULs = NULL

Lần lượt hội item d với các phần tử đứng phía sau d trong quan hệ thứ tự D < B < A < E < C để tạo ra tập item mới: { D,B }, { D,A },{ D,E },{ D,C }, tính UL cho các tập mới phát sinh này.

UL1({D,B}) = (30,15): ta có iutils + rutils = 30 + 15 = 45 40 đưa vào exULs.

UL2({D,A}) = (24,5): ta có iutils + rutils = 24 + 5 = 29 < 40 không đưa vào exULs.

UL3({D,E}) = (24,4): ta có iutils + rutils = 24 + 4 = 28 < 40 không đưa vào exULs.

UL4(D,C) = (33,0): ): ta có iutils + rutils = 33 + 0 = 33 < 40 không đưa vào exULs.

Do UL1({D,B}) = 45 > 40 nên ta đi xét ta đi xét tiếp các tập {D,B,A}, {D,B,E}, {D,B,C} như sau:

UL5({DBA}) = (21,4): ta có iutils + rutils = 21 + 4 = 25 < 40 không đưa vào exULs.

UL6({DBE}) = (36,4): ta có iutils + rutils = 36 + 4 = 40 40 đưa vào exULs.

UL7({DBC}) = (34,0): ta có iutils + rutils = 34 + 0 = 34 < 40 không đưa vào exULs.

Do UL6(DBE) = 40 minutil nên ta sẽ xét tiếp các tập phát sinh {DBEC} như sau:

| Tid | Iutils | Rutils |
| --- | --- | --- |
| T3 | 20 | 0 |
| T4 | 20 | 0 |
| Tổng | 40 | 0 |

UL7({DBEC) = (40,0): ta có iutils + rutils = 40 + 0 = 40 40 đưa vào exULs.

Kết thúc vì không còn tập phát sinh.

Ta xét danh mục B: có giá trị tổng iutils = 16, tổng giá trị Rutils = 20, ta có iutils + rutils = 16 + 20 = 36 < 40 nên không mở rộng các phần tử sau danh mục B.

Xét danh mục A: có giá trị tổng iutils = 20, tổng giá trị Rutils = 17, ta có iutils + rutils = 20 + 17 = 37 < 40 nên không mở rộng các phần tử sau danh mục A.

Xét danh mục E: có giá trị tổng iutils = 15, tổng giá trị Rutils = 12, ta có iutils + rutils = 15 + 12 = 27 < 40 nên không mở rộng các phần tử sau danh mục E.

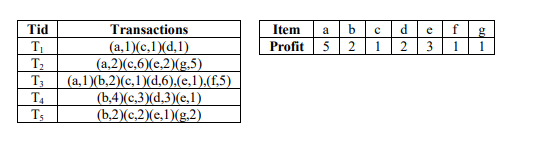
Xét danh mục C: có giá trị tổng iutils = 13, tổng giá trị Rutils = 0, ta có iutils + rutils = 13 + 0 = 13 < 40 nên không mở rộng các phần tử sau danh mục C.

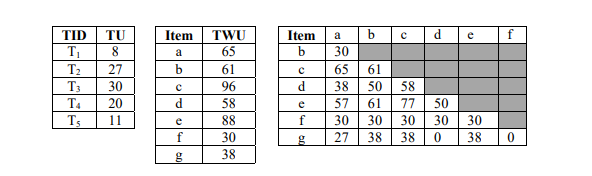
Vậy các tập: {D}, {DB}, {DBE}, {DBEC} đây là các mục tiện ích cao mà HUI-Miner khai thác được.

##### FHM (2014)

HUI-Miner là một thuật toán rất hiệu quả. Tuy nhiên, một nhược điểm là thao tác tham gia để tính toán danh sách tiện ích của một mặt hàng rất tốn kém. Thuật toán FHM cải thiện HUI-Miner bằng cách có thể loại bỏ các bộ mục tiện ích thấp mà không cần thực hiện các thao tác nối.

Sử dụng ví dụ trên thì ta có như sau:





TWU của tất cả các cặp mặt hàng và lưu trữ nó trong một cấu trúc có tên EUCS và có thể được triển khai dưới dạng (1) ma trận tam giác hoặc (2) hashmap của hashmap.

VD: twu({a,b}) = 30, twu({a,c})=65

Sau đó, trong quá trình tìm kiếm, hãy xem xét rằng chúng ta cần tính toán danh sách tiện ích của một mục X. Nếu X chứa một cặp mục i và j sao cho TWU ({i, j}) < minutil, thì X là tiện ích thấp cũng như tất cả các phần mở rộng của nó.

VD: Ta cần tính 1 mục mới {a,b,c}. mà ngướng minutil là 50. Vậy thì không cần tạo vì twu({a,b}) không thể lớn hơn 30

Do đó chỉ cần sử dụng bảng đơn giản này chúng ta chỉ cần thực hiện 1 lần bằng cách đọc cơ sở dữ liệu chúng ta có thể loại bỏ rất nhiều phép nối theo cách HUI

##### FHM+ (2016)

##### ULB-Miner(2018)

Ngoài ra còn 1 số thuật toán khác: HUP-Miner(2015), HMiner (2017),… mới nhất là HUIM-SU (2022)

##### 

1.3. Tổng quan về mạng nơ-ron học sâu

1.3.1 Mạng Nơ-ron nhân tạo

1.3.2. Mạng học sâu

1.3.3. Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN)

1.3.3.1. Graph hay đồ thị là gì?

- Kn

- Biểu đồ và nơi tìm thấy chúng

- Dữ liệu có cấu trúc biểu đồ xử lý những loại vấn đề nào?

- …..

1.3.3.2. Giới thiệu về GNN

1.3.3.3. Các thành phần chính của GNN

1.3.3.4. Cách thức hoạt động của GNN

1.

Chương 2: Thiết Kế Hệ Thống

2.1.

2.2.

Chương 3: Phát Triển và Triển Khai

3.1.

3.2.

Chương 4: Thực Nghiệm và Đánh Giá

4.1 Kết quả đạt được của đề tài

4.2 Hạn chế

4.3 Hướng phát triển

**Phần KẾT LUẬN**

Tài liệu tham khảo

[1]…

[2]…