Algorithm 3: Generate Sanitized Database

Trần Khắc Bình

05/08/2024

1 Tổng quan

Theo như paper, FULD được chia làm 3 thuật toán nhỏ:

- Algorithm 1: Xây dựng utility-list dictionary (UTLDic) từ transaction database (csdl gốc).
  Mã giả của thuật toán được trình bày trong paper như sau:

Algorithm 1 Construct UTLDic Algorithm

Require: the database D, the sensitive high-utility itemsets S, the non-sensitive high-utility itemsets NS
Ensure: the utility-list dictionary $UTLDic$

$SItem = \{\}$
for each $S_i \in S$ **do**
    $SItem = SItem \cup S_i$
end for
$SItem = set(SItem)$
$UTLDic = \phi$
for each $item \in SItem$ **do**
    create a new node $UTList$
    $UTList.item\_name = item$, $UTList.sum\_utility = 0$
    $UTList.ULElems = \phi$, $UTList.SINS = 0$
    calculate $SINS(item)$ according to Definition 14
    $UTLDic[item] = UTList$
end for
for each $T_i \in DB$ **do**
    calculate $tns(T_i)$ using Eq. c
    get sensitive items SI of T_i
    for each $item \in SI$ **do**
        create a new ULElem node ULE
        $ULE.TID =$ the TID of T_i, $ULE.utility = q(item, T_i) \times p(item)$
        $ULE.tns = tns(T_i)$
        $UTLDic[item].ULElems.append(ULE)$
        $UTLDic[item].sum\_utility += ULE.utility$
    end for
end for
return $UTLDic$

- Algorithm 2: Tạo ra sanitized UTLDic bằng cách ẩn sensitive high-utility itemsets dựa vào UTLDic. Mã giả của thuật toán được trình bày trong paper như sau:

Algorithm 2 Hide Sensitive High-utility Itemsets Algorithm

Require: $UTLDic$, min_util, the sensitive high-utility itemsets S, the database D
Ensure: the sanitized $UTLDic$

```
1: sort S in descending order of u(S_i) (S_i ∈ S)
2: for each S_i ∈ S do
3:     sort S_i in ascending order of SINS(item) (item ∈ S_i)
4:     calculate l = L(S_i) according to Definition 19
5:     targetUtil = u(S_i) − min_util + 1
6:     while targetUtil > 0 do
7:         for each item ∈ S_i do
8:             UTlist = UTLDic[item].UTList
9:             sort ULElems of UTlist order by tns desc, utility asc
10:            for each elem ∈ UTlist and elem ∈ l and targetUtil > 0 do
11:                if elem.utility ≤ targetUtil then
12:                    targetUtil− = elem.utility
13:                    elem.utility = 0
14:                else
15:                    count = q(item, elem.TID) − ⌈targetUtil / p(item)⌉
16:                    elem.utility = count × p(item)
17:                    targetUtil = 0
18:                end if
19:                update UTLDic[item].sum_utility
20:            end for
21:        end for
22:    end while
23: end for
```

- Algorithm 3: Tạo ra sanitized database (csdl đã được làm sạch) dựa vào sanitized UTLDic. (Không được trình bày và không có mã giả trong paper)

2 Algorithm 3: Generate Sanitized Database

Ý TƯỞNG:

Sau khi đã ẩn các sensitive high-utility itemsets thông qua việc điều chỉnh UTLDic, chúng ta cần sinh ra một cơ sở dữ liệu đã được làm sạch (sanitized database). Thuật toán Generate Sanitized Database thực hiện nhiệm vụ này bằng cách áp dụng các thay đổi từ UTLDic đã được sanitize vào bản sao của cơ sở dữ liệu gốc.

MÃ GIẢ:

Algorithm 3 Generate Sanitized Database

Require: $sanitized\_UTLDic$, the database D
Ensure: sanitized database $sanitized\_db$

$sanitized\_db \leftarrow copy(D)$
for each $(item, UTlist) \in sanitized\_UTLDic$ **do**
    for each $elem \in UTlist.ULElems$ **do**
        $tid \leftarrow elem.TID$
        $modified\_utility \leftarrow elem.item\_utility$
        $new\_internal\_utility \leftarrow \lfloor \frac{modified\_utility}{p(item)} \rfloor$
        if $new\_internal\_utility = 0$ **then**
            Remove $item$ from $sanitized\_db[tid]$
        else
            $sanitized\_db[tid][item] \leftarrow new\_internal\_utility$
        end if
    end for
end for
for each $transaction \in sanitized\_db$ **do**
    if $transaction$ is empty **then**
        Remove $transaction$ from $sanitized\_db$
    end if
end for
return $sanitized\_db$

GIẢI THÍCH THUẬT TOÁN:

1. Khởi tạo sanitized_db là một bản sao của cơ sở dữ liệu gốc.
2. Duyệt qua từng item trong UTLDic đã được làm sạch:
    - Với mỗi ULElem trong UTList của item:
        * Lấy TID và giá trị utility đã được điều chỉnh.
        * Tính toán internal utility mới dựa trên giá trị utility đã điều chỉnh và external utility của item.
        * Nếu internal utility mới bằng 0, loại bỏ item khỏi transaction tương ứng của sanitized_db.
        * Ngược lại, cập nhật internal utility của item trong transaction của sanitized_db.
3. Sau khi đã xử lý tất cả các item:
    - Duyệt qua tất cả các transaction trong sanitized_db.
    - Loại bỏ các transaction rỗng (nếu có).
4. Trả về cơ sở dữ liệu đã được làm sạch sanitized_db.

ĐỘ PHỨC TẠP:

- Thời gian: $O(n \times m)$, trong đó n là số lượng item trong UTLDic và m là số lượng ULElem trung bình trong mỗi UTList.
- Không gian: $O(|D|)$, trong đó $|D|$ là kích thước của cơ sở dữ liệu gốc.

Thuật toán này đảm bảo rằng sanitized database phản ánh tất cả các thay đổi đã được thực hiện trong quá trình ẩn các sensitive high-utility itemsets, đồng thời duy trì cấu trúc và tính nhất quán của original database.