

# 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  $UTLDic$

```
1:  $SItem = \{\}$ 
2: for each  $S_i \in S$  do
3:    $SItem = SItem \cup S_i$ 
4: end for
5:  $SItem = set(SItem)$ 
6:  $UTLDic = \phi$ 
7: for each  $item \in SItem$  do
8:   create a new node  $UTList$ 
9:    $UTList.item\_name = item, UTList.sum\_utility = 0$ 
10:   $UTList.ULElems = \phi, UTList.SINS = 0$ 
11:  calculate  $SINS(item)$  according to Definition 14
12:   $UTLDic[item] = UTList$ 
13: end for
14: for each  $T_i \in DB$  do
15:   calculate  $tns(T_i)$  using Eq. c
16:   get sensitive items  $SI$  of  $T_i$ 
17:   for each  $item \in SI$  do
18:     create a new ULElem node  $ULE$ 
19:      $ULE.TID =$  the TID of  $T_i, ULE.utility = q(item, T_i) \times p(item)$ 
20:      $ULE.tns = tns(T_i)$ 
21:      $UTLDic[item].ULElems.append(ULE)$ 
22:      $UTLDic[item].sum\_utility += ULE.utility$ 
23:   end for
24: end for
25: 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  $u(S_i)$  ( $S_i \in S$ )
2: for each  $S_i \in S$  do
3:   sort  $S_i$  in ascending order of  $SINS(item)$  ( $item \in S_i$ )
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  $item \in S_i$  do
8:        $UTlist = UTLDic[item].UTList$ 
9:       sort  $ULElems$  of  $UTlist$  order by  $tns$  desc,  $utility$  asc
10:      for each  $elem \in UTlist$  and  $elem \in l$  and  $targetUtil > 0$  do
11:        if  $elem.utility \leq targetUtil$  then
12:           $targetUtil - = elem.utility$ 
13:           $elem.utility = 0$ 
14:        else
15:           $count = q(item, elem.TID) - \lceil \frac{targetUtil}{p(item)} \rceil$ 
16:           $elem.utility = count \times p(item)$ 
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*

```
1: sanitized_db  $\leftarrow$  copy(D)
2: for each (item, UTlist)  $\in$  sanitized_UTLDic do
3:   for each elem  $\in$  UTlist.ULElems do
4:     tid  $\leftarrow$  elem.TID
5:     modified_utility  $\leftarrow$  elem.item_utility
6:     new_internal_utility  $\leftarrow \lfloor \frac{\text{modified\_utility}}{p(\text{item})} \rfloor$ 
7:     if new_internal_utility = 0 then
8:       Remove item from sanitized_db[tid]
9:     else
10:      sanitized_db[tid][item]  $\leftarrow$  new_internal_utility
11:    end if
12:  end for
13: end for
14: for each transaction  $\in$  sanitized_db do
15:   if transaction is empty then
16:     Remove transaction from sanitized_db
17:   end if
18: end for
19: 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.