HIGH UTILITY GRADUAL ITEMSETS MINING

FONGUE ASSONDJI Priscile Audrey

Computer Science student at University of Yaounde I
Field: Data Science

WoCC23, Polytechnique-Yaoundé, 2023

Overview

1. Introduction
2. Definitions
3. State of art
4. High utility gradual itemsets mining
5. Experimental results
6. Conclusion and outlook

Introduction

Accumulation of large volume of data

Extracting knowledge

Supermarkets

Profitability analysis

Definitions

Let $I = \{i_1, ..., i_n\}$ a set of items and $\Delta = \{T_1, ..., T_m\}$ a quantitative transaction database where $T_z \subseteq I$.

Gradual item (Lonlac et al., 2020)

It is an item in the form of i^*, from an attribute i and a variation $* \in \{\geq, \leq\}$ or $\{+, -\}$ of the values of i.

Gradual itemset or gradual pattern, (Lonlac et al.,2020)

It is a non-empty set of gradual items denoted by M $= \{i_1^{*1}, ..., i_k^{*k}\}$.

Support of a gradual itemset (Di-jorio et al.,2009; Lonlac et al.,2020)

Let M be a pattern with $\{L_1, ... L_m\}$.

Supp(M) $= \frac{Max_{1 \leq i \leq m}(|L_i|)}{|\Delta|}$

$\text{Sup}(\{i_1^{*1}, ..., i_k^{*k}\}, \Delta)) = \frac{\Delta(\{i_1^{*1}, ..., i_k^{*k}\})}{|\Delta|(|\Delta|-1)/2}$

Example

Hostel	Town	Pop.(10^3)	Dist. from centre	Price
h_1	Paris	2.1	0.3	82
h_2	New York	8.0	5	25
h_3	New York	8.0	0.2	135
h_4	Ocala	0.04	0.1	60

Table: Example of database [Di-jorio et al, 2009] (Δ)

minsup = 50%

Ordered database:

- $\{Dist^+\}$:(h_1, h_3)
- $\{Prix^+\}$:(h_1, h_3)
- $\{Prix^+ Dist^+\}$:(h_1, h_3)
- $\{Pop^+, Dist^-, Prix^+\}$: (h_1, h_3)

$(h_1[pop] \leq h_3[pop], h_1[Dist] \geq h_3[Dist], h_1[Prix] \leq h_3[Prix])$

Unordered database:

- $\{Pop^+, Prix^+\}$: (h_4, h_1, T_3)

Concept of utility

TID	Transaction
T_0	$(a,1),(b,5),(c,1),(d,3),(e,1)$
T_1	$(b,4),(c,3),(d,3),(e,1)$
T_2	$(a,1),(c,1),(d,1)$
T_3	$(a,2),(c,6),(e,2)$
T_4	$(b,2),(c,2),(e,1)$

Table: Quantitative database

Item	External utility value
a	5
b	2
c	1
d	2
e	3

Table: External utility values

Concept of utility (Liu et al., 2005)

D: Quantitative database; I: set of items; $T_c \subseteq$ I

Utility of an item, Utility of an itemset

- $u(i, T_c) = \mathtt{p(i)} * \mathtt{q(i,}T_c\mathtt{)}$
- $u(i,D) = \sum_{T_c \in g(i)} u(i, T_c)$
- $u(X, T_c) = \sum_{i \in X} \mathtt{u(i,}T_c\mathtt{)}$
- $u(X,D) = \sum_{T_c \in g(X)} \mathtt{u(X,}T_c\mathtt{)}$

Concept of utility (Liu et al., 2005)

D: Quantitative database; I: set of items; $T_c \subseteq$ I

Utility of an item, Utility of an itemset

- $\mathrm{u}(\mathrm{i}, T_c) = \mathtt{p(i)} * \mathtt{q(i,}T_c\mathtt{)}$
- $\mathrm{u}(\mathrm{i},\mathrm{D}) = \sum_{T_c \in g(i)} \mathrm{u}(\mathrm{i}, T_c)$
- $\mathrm{u}(\mathrm{X}, T_c) = \sum_{i \in X} \mathtt{u(i,}T_c\mathtt{)}$
- $\mathrm{u}(\mathrm{X},\mathrm{D}) = \sum_{T_c \in g(X)} \mathtt{u(X,}T_c\mathtt{)}$

High utility itemset

An itemset (a pattern) **X** is said to be a *high utility itemset* iff `u(X,D)` $\geq$ **minutil**

TWU(Transaction-weighted utilization)

- $\mathrm{TU}(T_c) = \sum_{x \in T_c} u(x, T_c)$
- $\mathrm{TWU}(\mathrm{X}) = \sum_{T_c \in g(X)} TU(T_c)$

Example

Utility calculation

- $u(a, T_2) = 5 * 1 = 5$
- $u(\{a,c\}, T_2) = u(a, T_2) + u(c, T_2) = 5 * 1 + 1 * 1 = 6$
- $u(\{a,c\},D) = u(\{a,c\}, T_0) + u(\{a,c\}, T_2) + u(\{a,c\}, T_3) = u(a, T_0) + u(c, T_0) + u(a, T_2) + u(c, T_2) + u(a, T_3) + u(c, T_3) = 5*1 + 1*1 + 5*1 + 1*1 + 5*2 + 1*6 = 28$

Transaction-weighted utilization, Transaction utility

- $TU(T_1) = 2*4 + 1*3 + 2*3 + 3*1 = 20$
- $TWU(\{c, d\}) = TU(T_0) + TU(T_1) + TU(T_2) = 25 + 20 + 8 = 53$

Applications: Retail, Finance.

Gradual itemset mining algorithms

- Algorithms consider different semantics of graduality (temporality, seasonality, emergence, etc...) depending on the data model to extract other variants of gradual itemsets.
- Advantage: suitable for real world applications where quantitative data are used
- Limit : They are inadequate for searching gradual itemsets that generate a high profit.

High utility itemsets algorithms

- There are two phases and one phase algorithms to extract high utility itemsets
- Advantage: allow the expression of other interests of the user on the patterns
- Limit: Most of the algorithms does not handle itemsets with negative item values from a large database.

How to offer the user itemsets whose utility varies considerably with time?

High utility gradual itemsets mining

Problem definition

" *Given a quantitative database and external utilities [Zida et al, 2017] of the different attributes, find gradual itemsets whose utility is higher than a utility threshold*" .

High utility gradual itemset

$$U(I,\Delta) = \sum_{(T_x,T_y)\in\Delta(I)\wedge q(i,T_x)\neq q(i,T_y)\forall i\in I}[U(I,T_y) - U(I,T_x)]$$

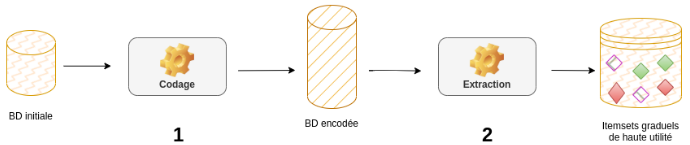


Figure: General scheme for extracting high-utility gradual patterns

Example of database

a:: Thon; b:: Tilapia; c:: Carp

$Tid \mid item$	a	b	c
T_1	3	7	4
T_2	2	5	8
T_3	4	5	2
T_4	1	6	9

Table: Quantitative database (D)

$Tid \mid item$	a	b	c
(T_1, T_2)	-1	-2	4
(T_1, T_3)	1	-2	-2
(T_1, T_4)	-2	-1	5
(T_2, T_3)	2	0	-6
(T_2, T_4)	-1	1	1
(T_3, T_4)	-3	1	7

Table: encoded database (D')[a]

[a] we are in temporality context

Approach 1 : extraction with separation of positive and negative (HUGI-Merging)

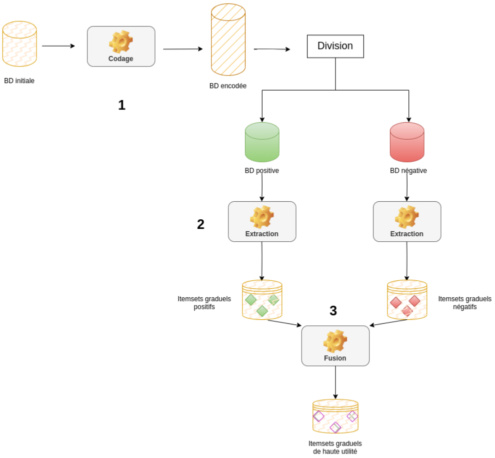


Figure: Process of extracting gradual high utility itemsets with partition of the

Example of extraction

D'	Transactions
1	(a-, -1); (b-, -2); (c+, 4)
2	(a+, 1); (b-, -2); (c-, -2)
3	(a-, -2); (b-, -1); (c+,5)
4	(a+, 2); (b, 0); (c-, -6)
5	(a-, -1); (b+, 1); (c+, 1)
6	(a-, -3); (b+, 1); (c+, 7)

D'1	Transactions
1	(c+, 4)
2	(a+, 1)
3	(c+, 5)
4	(a+, 2)
5	(b+, 1), (c+, 1)
6	(b+, 1), (c+, 7)

D'2	Transactions
1	(a-, -1), (b-, -2)
2	(b-, -2), (c-, -2)
3	(a-, -2), (b-, -1)
4	(c-, -6)
5	(a-, -1)
6	(a-, -3)

Figure: Division of the DB: D'1, D'2

Exemple of extraction

In D'2, we obtain:

- $U(\{a-\}) = -7 \rightarrow$ TID: [1,3,5,6]
- $U(\{c-\}) = -8 \rightarrow$ TID: [2,4]
- $U(\{a-,b-\}) = -6 \rightarrow$: [1,3]

In D'1, we obtain:

- $U(\{c+\}) = 17 \rightarrow$ TID: [1,3,5,6]
- $U(\{b+, c+\}) = 10 \rightarrow$ TID: [5,6]

After merging, we obtain:

- $\mathbf{U(\{a-, c+\}) = 10}$
- $\mathbf{U(\{ a-, b+, c+\}) = 6}$
- $U(\{a-, b-, c+\}) = 3$

$U(\{\mathbf{b-\ c+}\}) = 6$

Approach 2: Extraction with a single database (HUGI)

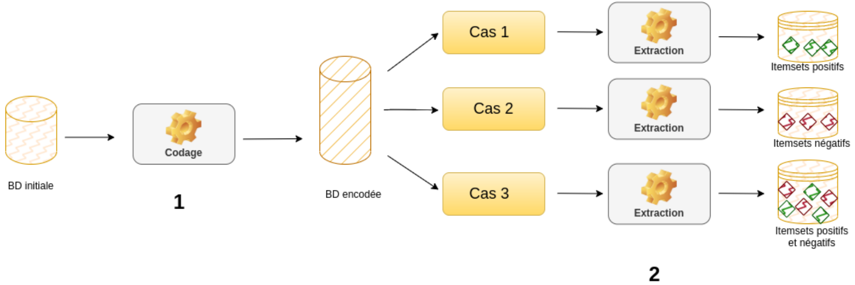


Figure: Process of extracting gradual high utility itemsets with a single database

The obtained itemsets

U (X) $\geq$ minutil.

itemset	Utility
b+ a- c+	6
b+ c+	10
a- c+	10
b- c+	6
c+	17

Table: mixed positive itemsets (case 1)

U (X) $\leq$ -minutil

itemset	Utility
c-	-8
b- a-	-6
a-	-7

Table: mixed negative itemsets (case 2)

U (X) $\leq$ -minutil or U (X) $\geq$ minutil

itemset	Utility
c-	-8
b+ a- c+	6
b+ c+	10
b- a-	-6
b- c+	6
a-	-7
a- c+	10
c+	17

Table: Gradual high utility itemsets (case 3)

Example of real high utility gradual itemsets

Dataset

- Domain : supermarket
- Number of instances : 1000
- Number of items : 45
- minUtil: 60000
- Number of patterns mined: 7 patterns

itemset	utility	itemset	utility
$\{Breakfastfoods^{\geq}, SideDishes^{\geq}, Dairy^{\geq}, snackFoods^{\geq}\}$	60125	$\{Candy^{\geq}\}$	60680
$\{BakingGoods^{\leq}\}$	-64900	$\{Hygiene^{\geq}\}$	61220
$\{Dairy^{\geq}, Snackfoods^{\geq}\}$	61806	$\{Dairy^{\geq}\}$	83205

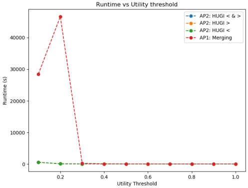
Figure: Patterns extracted

Data and working environment

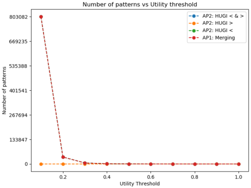
- Dataset: **Order** $->$ 34555,16383
- External utilities: between 5 et 20
- Environment: intel Core i7-8750H 2.2GHz CPU, 8Go of RAM, processor 12 cores.
- Utility threshold: from 0.1 to 1
- Librairies: Efficient_apriori, numpy, Matplotlib, Subprocess, pandas

- *HUGI$>$*
- *HUGI$<$*
- *HUGI$<$ & $>$*
- *HUGI-Merging*
- *T-Gpatterns*

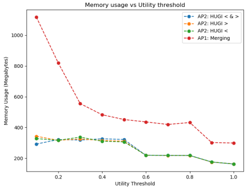
Experiment 1



Time vs minUtil



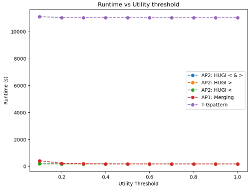
Number of patterns vs minUtil



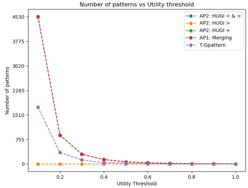
Memory usage vs minUtil

Figure: Comparative evaluation of HUGI et HUGI-Merging sur le dataset *Order* (100 items, 418 transactions)

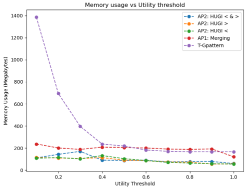
Experiment 2



Time vs minutil



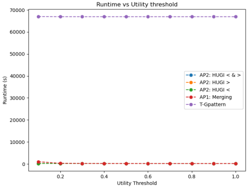
Number of patterns vs minUtil



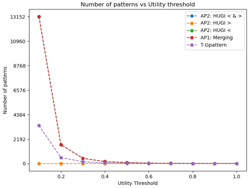
Memory usage vs minUtil

Figure: Comparative evaluation of HUGI et HUGI-Merging sur le dataset *Order* (40 items, 418 transactions)

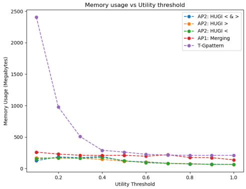
Experiment 3



Time vs minUtil



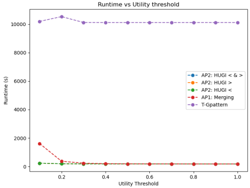
Number of patterns vs minUtil



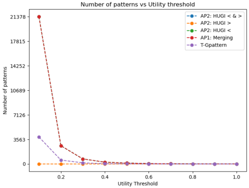
Memory usage vs minUtil

Figure: Comparative evaluation of HUGI and HUGI-Merging on the dataset *Order* (50 items, 418 transactions)

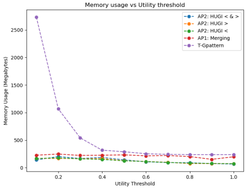
Experiment 4



Time vs minUtil



Number of patterns vs minUtil



Memory usage vs minUtil

Figure: Comparative evaluation of HUGI et HUGI-Merging sur le dataset *Order* (60 items, 418 transactions)

Conclusion

- Search for high-utility gradual patterns
- Use of an existing high-utility pattern mining algorithm
- Proposal of two high-utility gradual pattern mining algorithms: *HUGI* and *HUGI-Merging*.
- Experimentation on a real data set

Outlook

- Combine support threshold and utility threshold to extract frequent gradual patterns of high utility
- Using idea to make recommendation
- Find a complete merging algorithm
- Find measures to assess the quality of extracted patterns
- Improve execution time on larger datasets

Some references

Lisa Di jorio,Anne Laurent,Maguelonne Teisseire (2009)
Mining Frequent Gradual itemsets from large databases
International Symposium on Intelligent Data Analysis, 297 – 308.

Jerry Lonlac, Engelbert Mephu Nguifo(2020)
A novel Algorithm for searching frequent gradual patterns from an ordered dataset
Intelligent Data Analysis 24(5), 1029 – 1042.

Philippe Fournier-Viger, Jerry Chun-wei Lin, Tin Truong-chi, Roger Nkambou (2019)
A Survey of High Utility Itemset Mining
High Utility Pattern Mining, 1 – 45.

Souleymane Zida , Philippe Fournier-Viger , Jerry Chun-Wei Lin ,Cheng-Wei Wu , Vincent S. Tseng (2017)
EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining
Advances in Artificial Intelligence and Soft Computing, 530 – 546.

Thanks for your great attention !