Mining and ranking closed itemsets from large-scale transactional datasets

Martin Kirchgessner

Laboratoire d'Informatique de Grenoble
martin.kirchgessner@imag.fr

September 26, 2016

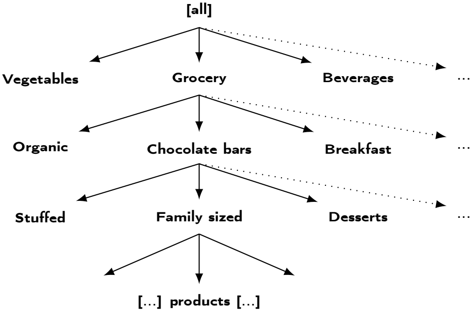
Outline

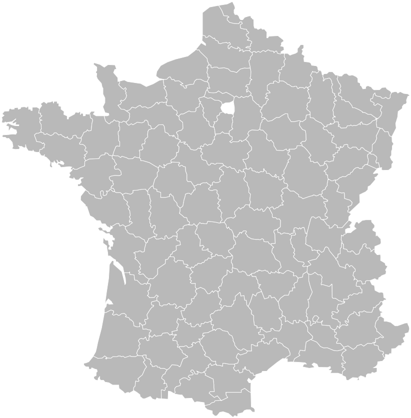
Over year 2013 at Intermarché

One of the major food stores in France

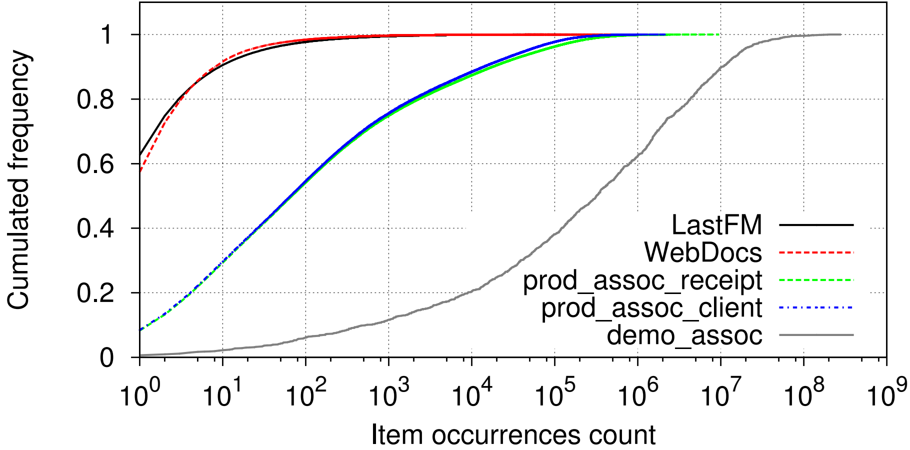
- 9,267,961 customers
- 290,734,163 tickets
- 222,228 different products

Additional data





Items distribution in our experimental datasets.

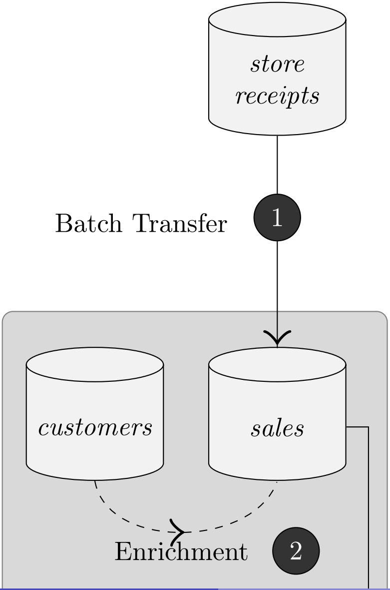


Interesting trends - 3 scenarios

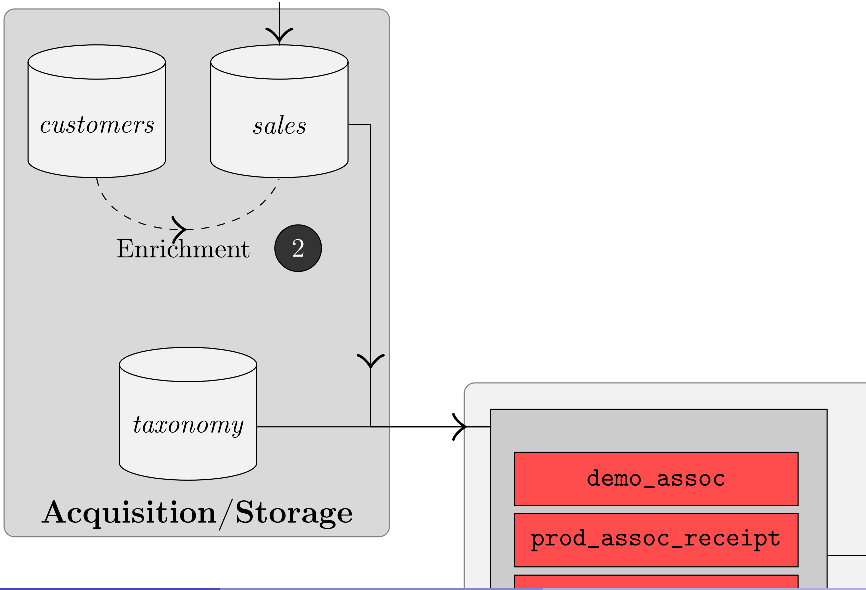
Final users: central marketing analysts.

- `demo_assoc`
  Women below 35 y.o. tend to buy baby food
  People from the Nord *department tend to buy sodas*
- `prod_assoc_client` : per-client products associations
  People who ever baught *vanilla cream* also bought *chocolate cream*
- `prod_assoc_receipt` : per-ticket products associations
  Pork sausage and *mustard* are often baught simultaneously with *dry Riesling*

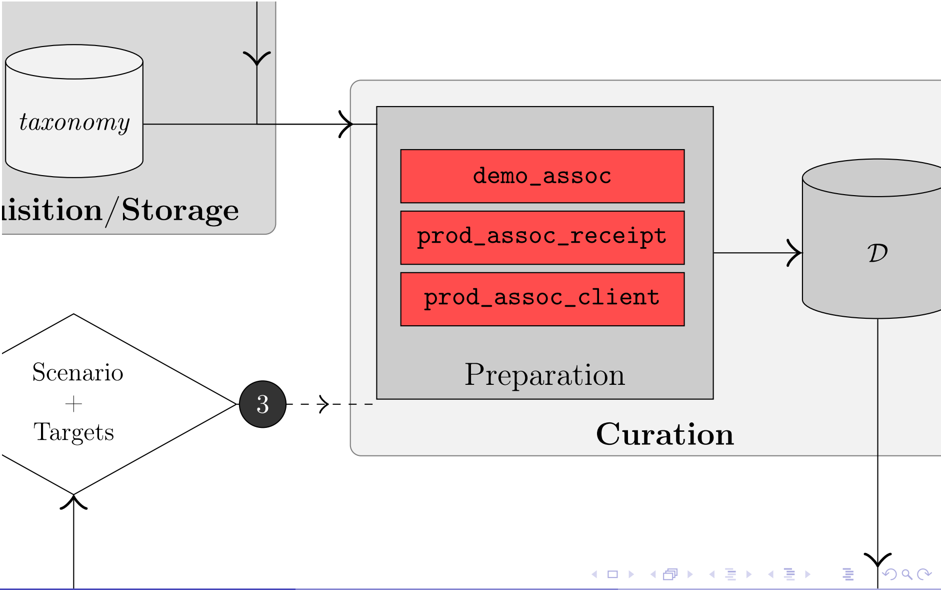
System overview



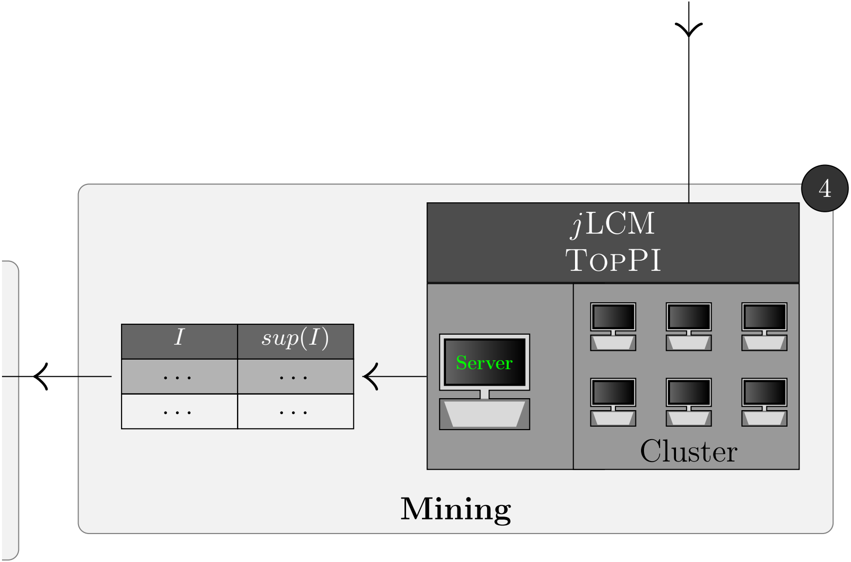
System overview



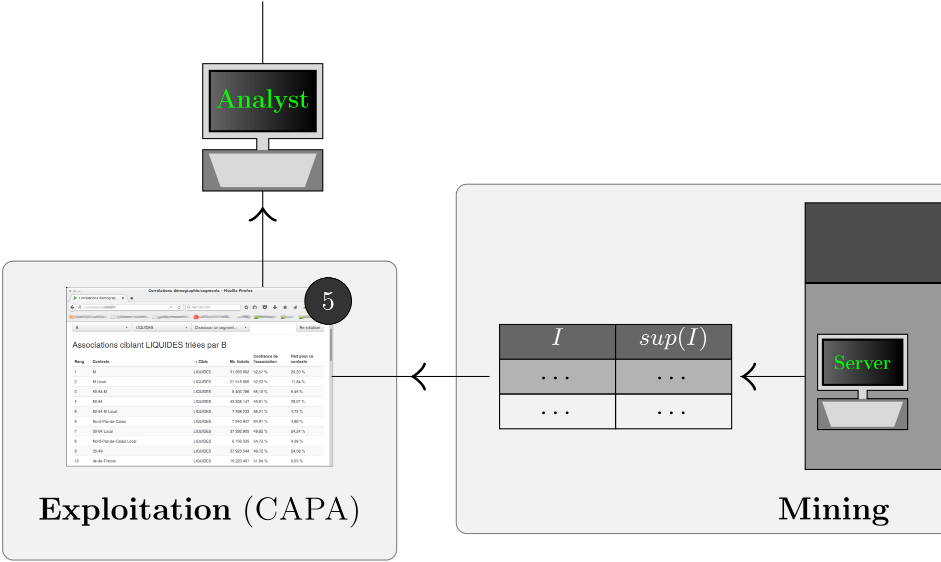
System overview



System overview



System overview



System overview

1. Nightly tickets transfer to the central store
2. Enrichment with taxonomy/demographics (HBase)
3. Curating $\mathcal{D}$: mining scenarios and target definition
    - Analyst may request any form of association rule
    - Over any set of target products/categories/populations
4. Association rules mining
5. Rules ranking and exploration

   TODO: pause and highlight Mining and Ranking

Transactional datasets

Input

Given $\mathcal{I}$, a set of items.
A collection $\mathcal{D}$ of *transactions* $\langle t_1, \ldots, t_n \rangle$, where each $t_j \subseteq \mathcal{I}$.

Transactional datasets

Input

Given $\mathcal{I}$, a set of items.
A collection $\mathcal{D}$ of *transactions* $\langle t_1, ..., t_n \rangle$, where each $t_j \subseteq \mathcal{I}$.

Output (presented to the analyst)

A collection of *closed* itemsets (CIS),
ie. itemsets P satisfying $\nexists Q \supset P$ s.t. $support_{\mathcal{D}}(P) = support_{\mathcal{D}}(Q)$.

Where $support_{\mathcal{D}}(P) = |\{t \in \mathcal{D} | P \subset t\}|$.

[12] *Discovering frequent closed itemsets for association rules*,
Pasquier, Bastide, Taouil, Lakhal @ ICDT'99

Frequency-based item ordering

Internally, items are represented as integers, indexed by decreasing frequency:

- 0 is the most frequent item
- 1 the second most
- etc...

Item-Centric Mining?

Big transactional datasets

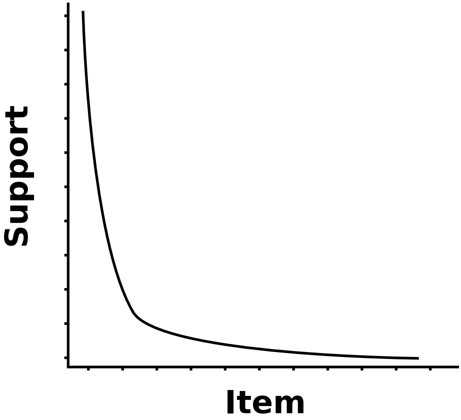
"big" means our datasets contain at least

- Thousands/millions of items in $\mathcal{I}$
- Millions of transactions in $\mathcal{D}$

Big transactional datasets

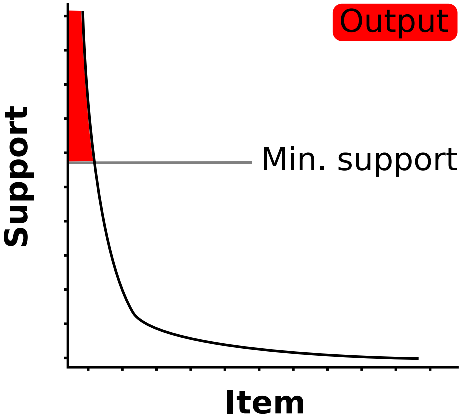
"big" means our datasets contain at least

- Thousands/millions of items in $\mathcal{I}$
- Millions of transactions in $\mathcal{D}$



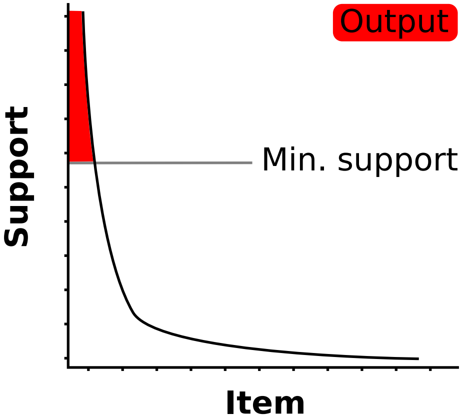
[2] *The Long Tail: Why the Future of Business Is Selling Less of More*, Anderson (2006)

Frequent Itemset Mining on big datasets



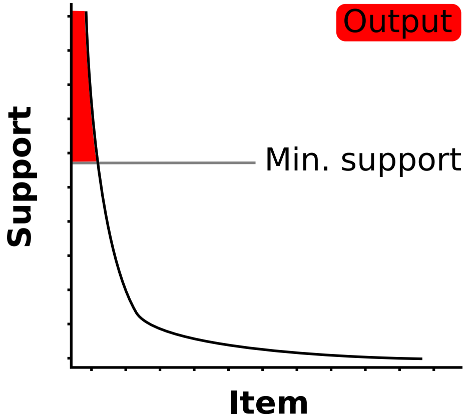
- Which minimum support yields interesting results?

Frequent Itemset Mining on big datasets



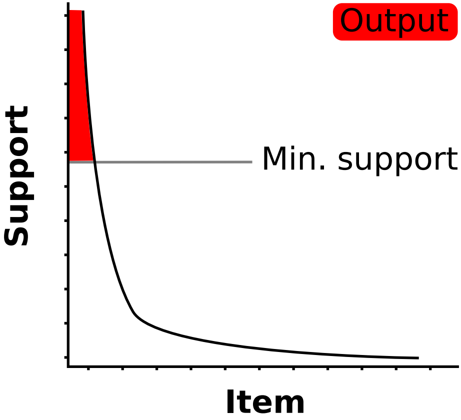
- Which minimum support yields interesting results?
- Are all closed itemsets interesting?

Frequent Itemset Mining on big datasets

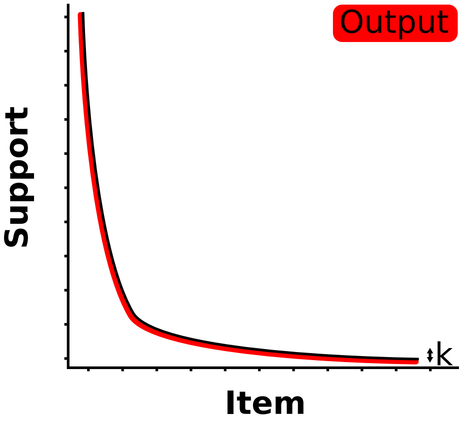


- Which minimum support yields interesting results?
- Are all closed itemsets interesting?
- What about the remaining items?

Item-Centric Mining



Item-Centric Mining



Replace the minimum support by a single parameter, k

Item-Centric Mining

TopPI 's problem statement

Given a transactional dataset $\mathcal{D}$ and an integer k,
return, $\forall i \in \mathcal{I}$, $top(i)$: the k most frequent CIS containing i.

TopPI stands for "Top **P**er **I**tem".

Item-Centric Mining

TopPI 's problem statement

Given a transactional dataset $\mathcal{D}$ and an integer k,
return, $\forall i \in \mathcal{I}$, $top(i)$: the k most frequent CIS containing i.

TopPI stands for "Top **P**er **I**tem".

Benefits

- Restrict intuitively the CIS space

Item-Centric Mining

TopPI 's problem statement

Given a transactional dataset $\mathcal{D}$ and an integer k,
return, $\forall i \in \mathcal{I}$, $top(i)$: the k most frequent CIS containing i.

TopPI stands for "Top **P**er **I**tem".

Benefits

- Restrict intuitively the CIS space
- Resulting collection is easy to browse

Item-Centric Mining

TopPI 's problem statement

Given a transactional dataset $\mathcal{D}$ and an integer k,
return, $\forall i \in \mathcal{I}$, $top(i)$: the k most frequent CIS containing i.

TopPI stands for "Top **P**er **I**tem".

Benefits

- Restrict intuitively the CIS space
- Resulting collection is easy to browse

We target high-end, multi-core servers.

Related Work

Can we implement Item-Centric Mining using existing methods ?

Our baseline: Item-Centric Mining with TFP

Implementation with a top-k CIS miner, TFP

For each item i:

- Instantiate $\mathcal{D}[i] = \{t \in \mathcal{D} | i \in t\}$
- Launch TFP on $\mathcal{D}[i]$, yielding $top(i)$.

[6] *Mining top-k frequent closed patterns without minimum support.*
Han, Wang, Lu, Tzvetkov @ ICDM'02

Our baseline: Item-Centric Mining with TFP

Implementation with a top-k CIS miner, TFP

For each item i:

- Instantiate $\mathcal{D}[i] = \{t \in \mathcal{D} | i \in t\}$
- Launch TFP on $\mathcal{D}[i]$, yielding $top(i)$.

Easy to parallelize, fine for small files.

[6] *Mining top-k frequent closed patterns without minimum support.*
Han, Wang, Lu, Tzvetkov @ ICDM'02

Our baseline: Item-Centric Mining with TFP

Implementation with a top-k CIS miner, TFP

For each item i:

- Instantiate $\mathcal{D}[i] = \{t \in \mathcal{D} | i \in t\}$
- Launch TFP on $\mathcal{D}[i]$, yielding $top(i)$.

Easy to parallelize, fine for small files.

Not sufficient for our datasets

Even with ad-hoc optimizations:

- Keep only top-k-frequent items in $\mathcal{D}[i]$
- Index transactions by item for an instant access to $\mathcal{D}[i]$.

[6] *Mining top-k frequent closed patterns without minimum support.*
Han, Wang, Lu, Tzvetkov @ ICDM'02

PFP: parallel FP-Growth

- An algorithm for the MapReduce platform.
- Returns, $\forall i \in \mathcal{I}$, at most k itemsets containing i.

[9] *PFP: parallel FP-growth for query recommendation.*
Li, Wang, Zhang, Zhang, Chang @ RecSys'08

PFP: parallel FP-Growth

- An algorithm for the MapReduce platform.
- Returns, $\forall i \in \mathcal{I}$, at most k itemsets containing i.
- Implementation available in (old versions of) Mahout.
  - Much more resource-consuming than TopPI and its baseline.

[9] *PFP: parallel FP-growth for query recommendation.*
Li, Wang, Zhang, Zhang, Chang @ RecSys'08

Efficiently enumerating CIS

TopPI has to find closed itemsets (CIS) and their support, but only those likely to appear in $top(i)$ for an item i.

Efficiently enumerating CIS

TopPI has to find closed itemsets (CIS) and their support, but only those likely to appear in $top(i)$ for an item i.

Enumeration is inspired from PLCM.

[11] *Discovering closed frequent itemsets on multi-core: Parallelizing computations and optimizing memory accesses.*

Négrevergne, Termier, Méhaut, Uno @ HPCS'10

Efficiently enumerating CIS

TopPI has to find closed itemsets (CIS) and their support, but only those likely to appear in $top(i)$ for an item i.

Enumeration is inspired from PLCM.

(P)LCM shapes the CIS lattice as a tree (depth-first traversal).

Tree property

In a branch, all itemsets P have the same $max(P)$.

[11] *Discovering closed frequent itemsets on multi-core: Parallelizing computations and optimizing memory accesses.*

Négrevergne, Termier, Méhaut, Uno @ HPCS'10

Frequency-based item ordering

- 0 is the most frequent item
- 1 the second most
- etc...

Frequency-based item ordering

- 0 is the most frequent item
- 1 the second most
- etc...

In a branch, an item is combined with items which are more frequent (globally).

The $top(i)$ heaps are firstly filled for the most frequent items.

TopPI 's main program

1. Instantiate a k-heap $top(i), \forall i$
2. Progressively fill them by enumerating CIS...

TopPI 's main program

1. Instantiate a k-heap $top(i), \forall i$
2. Progressively fill them by enumerating CIS... **and prune the enumeration when the concerned items already have a complete $top(i)$.**

TopPI 's main program

1. Instantiate a k-heap $top(i), \forall i$
2. Progressively fill them by enumerating CIS... **and prune the enumeration when the concerned items already have a complete $top(i)$.**

We can poll each item's heap via $min(top(i))$: the smallest itemset support in $top(i)$.

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1				
2				
$min(top(i))$	2	2	2	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$			
2				
$min(top(i))$	2	2	2	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$		
2				
$min(top(i))$	2	2	2	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$		
2				
$min(top(i))$	2	2	2	2

$\{0, 1\}$ is not closed

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	
2				
$min(top(i))$	2	2	2	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	
$min(top(i))$	4	2	4	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	
$min(top(i))$	4	2	4	2

$\{1, 2\}$ is not closed

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	
$min(top(i))$	4	2	4	2

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	$\{0, 3\}, 4$
$min(top(i))$	4	2	4	4

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	$\{0, 3\}, 4$
$min(top(i))$	4	2	4	4

$\{1, 3\}$ is not closed

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0, 2\}, 4$		$\{0, 2\}, 4$	$\{0, 3\}, 4$
$min(top(i))$	4	2	4	4

$support_{\mathcal{D}[3]}(2) = 4$, can we prune $\{2, 3\}$?

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0,2\}, 4$		$\{0,2\}, 4$	$\{0,3\}, 4$
$min(top(i))$	4	2	4	4

$support_{\mathcal{D}[\{2,3\}]}(0) = 3$, can we prune $\{0, 2, 3\}$?

An example (Fig.4.3, p.42)

k	$top(0)$	$top(1)$	$top(2)$	$top(3)$
1	$\{0\}, 6$	$\{1\}, 5$	$\{2\}, 5$	$\{3\}, 5$
2	$\{0, 2\}, 4$	$\{0, 1, 2, 3\}, 2$	$\{0, 2\}, 4$	$\{0, 3\}, 4$
$min(top(i))$	4	2	4	4

An example

After enumerating $\{c, d\}(support = 100)$
$\rightarrow$ we try to insert it in $top(c)$ and $top(d)$.

An example

After enumerating $\{c, d\}$($support = 100$)
$\rightarrow$ we try to insert it in $top(c)$ and $top(d)$.

Then, before attempting to find $\{b, c, d\}$

- we know that $support_{\mathcal{D}}(\{b, c, d\}) \leq 100$
- Can we prune if $top(b)$, $top(c)$ and $top(d)$ already have k CIS of support ≥ 100?
  ie. $min(top(b)) \geq 100$, idem for c and d.

An example

After enumerating $\{c, d\}(support = 100)$
$\rightarrow$ we try to insert it in $top(c)$ and $top(d)$.

Then, before attempting to find $\{b, c, d\}$

- we know that $support_{\mathcal{D}}(\{b, c, d\}) \leq 100$
- Can we prune if $top(b)$, $top(c)$ and $top(d)$ already have k CIS of support ≥ 100?
  ie. $min(top(b)) \geq 100$, idem for c and d.

Deeper in the enumeration...

Pruning $\{b, c, d\}$ implies to prune $\{a, b, c, d\}$.
Maybe $\{a, b, c, d\}$ is a relevant result for $top(a)$!

If $min(top(a)) \leq 100$, we cannot prune $\{b, c, d\}$.

TopPI 's challenges

- guarantee the completeness of all *top*(i)

TopPI 's challenges

- guarantee the completeness of all $top(i)$
- evaluate *quickly* if the enumeration of some CIS can be avoided
  - How many $min(top(i))$ invocations to decide (correctly) to prune?

TopPI 's challenges

- guarantee the completeness of all $top(i)$
- evaluate *quickly* if the enumeration of some CIS can be avoided
  - How many $min(top(i))$ invocations to decide (correctly) to prune?
- filter intermediate datasets without a user-provided minimum support

Pruning in TopPI

In a sub-branch rooted at an itemset P,
all closed itemsets Q will verify:

- $max(Q) = max(P)$
- $support_{\mathcal{D}}(Q) \leq support_{\mathcal{D}}(P)$

TopPI 's basic pruning principle

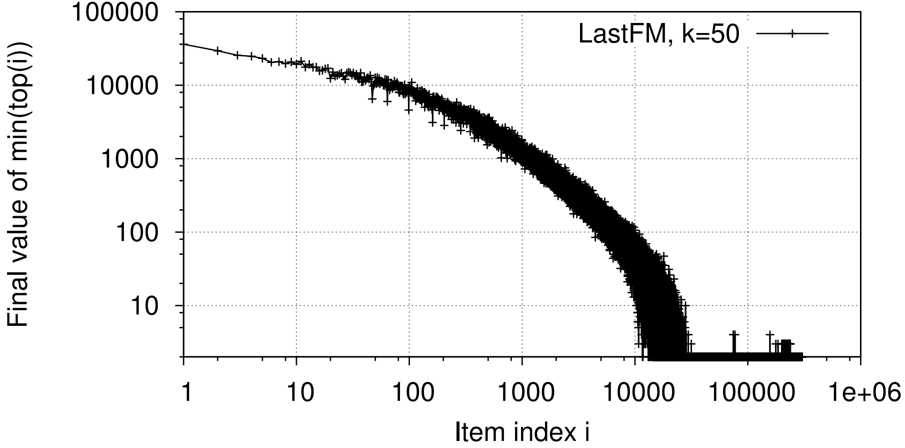
If, $\forall i < max(P), min(top(i)) \geq support_{\mathcal{D}}(P)$,
then the branch rooted at P can be pruned.

Deciding quickly to prune with prefix short-cutting

A rigorous pruning requires testing $min(top(i)), \forall i < max(P), \forall P$.

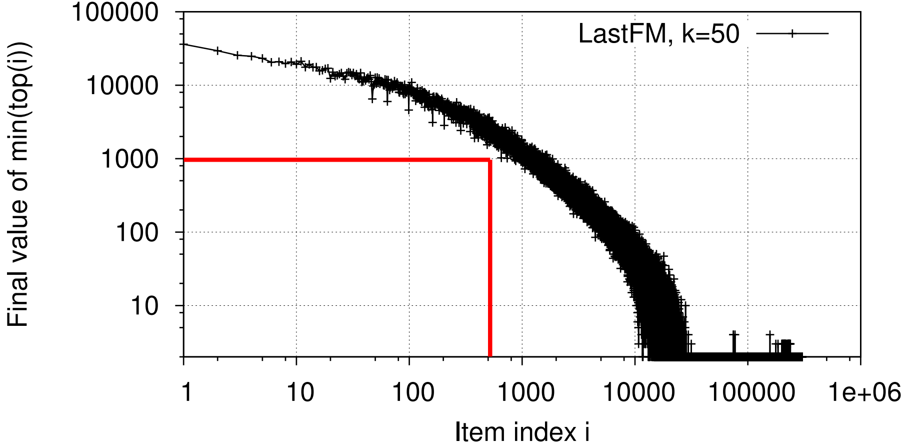
Deciding quickly to prune with prefix short-cutting

A rigorous pruning requires testing $min(top(i)), \forall i < max(P), \forall P$.



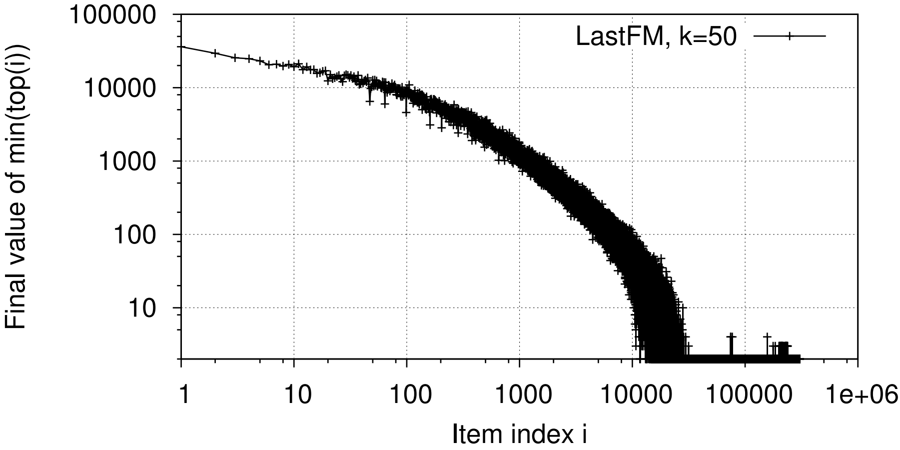
Deciding quickly to prune with prefix short-cutting

A rigorous pruning requires testing $min(top(i)), \forall i < max(P), \forall P$.

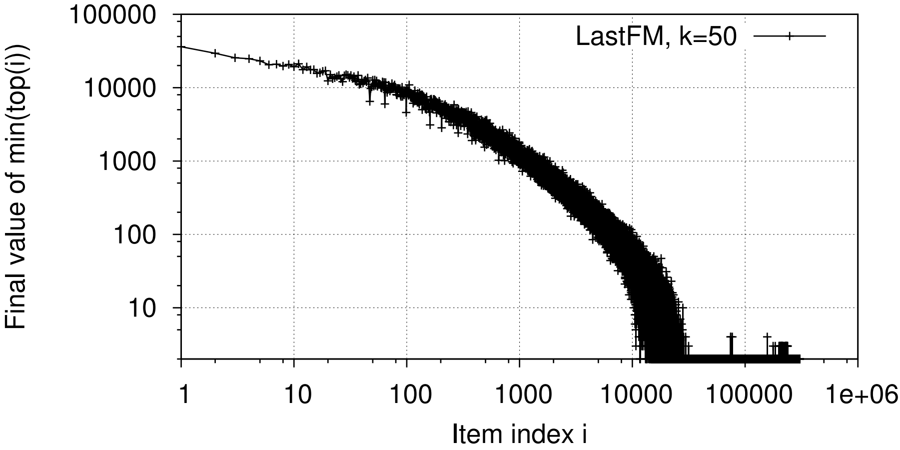


Here if $support_{\mathcal{D}}(P) \leq 1000$, no need to test $min(top(i))$ for $i < 500$.

Dynamic threshold adjustment



Dynamic threshold adjustment



Dynamic threshold adjustment

Finding a minimum frequency threshold adapted to each CIS branch.

Parallelization

As in PLCM, we dispatch each CIS branch between threads.
$\rightarrow$ excellent speed-up.

Parallelization

As in PLCM, we dispatch each CIS branch between threads.
$\rightarrow$ excellent speed-up.

Concurrent accesses in TopPI

Most access to the *top* collections are *read* accesses.

Two experiments

1. **Baseline comparison**
   apply a top-k CIS miner on each item's supporting transactions.

2. **Individual impact of our contributions**
   by disabling each one.

Experiments set-up

Datasets

Dataset	$\|\mathcal{I}\|$	$\|\mathcal{D}\|$	File size
Tickets	$222,228$	$290,734,163$	24GB
Clients	$222,228$	$9,267,961$	13.3GB
LastFM	$1,206,195$	$1,218,831$	277MB

Experiments set-up

Datasets

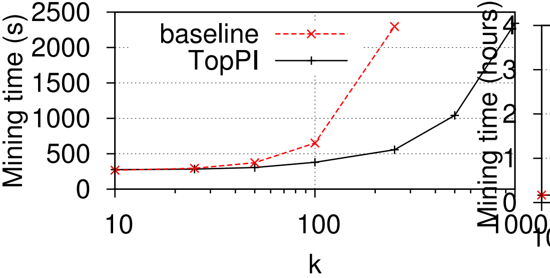
Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{D}\rvert$	File size
Tickets	$222,228$	$290,734,163$	24GB
Clients	$222,228$	$9,267,961$	13.3GB
LastFM	$1,206,195$	$1,218,831$	277MB

We measure run-times

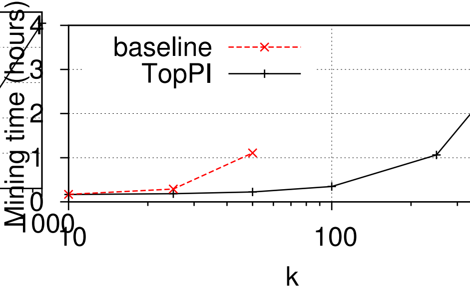
- Averaged over 3 attempts
- Not including the time to load $\mathcal{D}$.
- On a single server:
  - 2 Intel Xeon E5-2650, providing 16 cores with Hyper Threading
  - 128GB of RAM

All programs are implemented in Java.

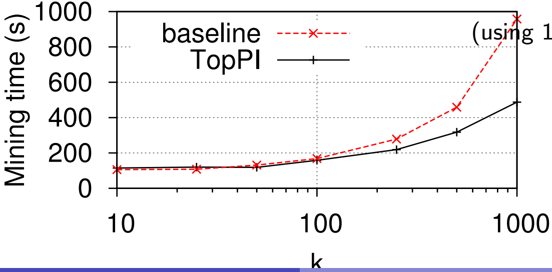
TopPI and Baseline run-times



prod_assoc_receipt



prod_assoc_client



(using 16 threads)

Contributions Impact

Dataset	TopPI
`prod_assoc_receipt`	222 s.
`prod_assoc_client`	661 s.
LastFM	116 s.

TopPI run-times (in seconds), using 32 threads and $k = 50$.

Contributions Impact

Dataset	TopPI	Without 3.5
prod_assoc_receipt	222 s.	1136 ($\times 5$)
prod_assoc_client	661 s.	Out of mem.
LastFM	116 s.	177 (+53%)

TopPI run-times (in seconds), using 32 threads and $k = 50$.

Section 4.2.5: Dynamic threshold adjustment

Contributions Impact

Dataset	TopPI	Without 3.5	Without 3.6
`prod_assoc_receipt`	222 s.	1136 ($\times$5)	230 (+4%)
`prod_assoc_client`	661 s.	Out of mem.	4177 ($\times$6)
LastFM	116 s.	177 (+53%)	150 (+29%)

TopPI run-times (in seconds), using 32 threads and $k = 50$.

Section 4.2.5: Dynamic threshold adjustment
Section 4.2.4: Pruning with prefix short-cutting

Contributions Impact

Dataset	TopPI	Without 3.5	Without 3.6	Without bo
`prod_assoc_receipt`	222 s.	1136 ($\times 5$)	230 (+4%)	3.8 hours, $\times$
`prod_assoc_client`	661 s.	Out of mem.	4177 ($\times 6$)	Out of mem
LastFM	116 s.	177 (+53%)	150 (+29%)	243 ($\times$2

TopPI run-times (in seconds), using 32 threads and $k = 50$.

Section 4.2.5: Dynamic threshold adjustment
Section 4.2.4: Pruning with prefix short-cutting

An example on retail data

Our `prod_assoc_receipt` dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

An example on retail data

Our `prod_assoc_receipt` dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

- $14,887$ ($< 0.005\%$) of these tickets contain "sushi rice"

An example on retail data

Our `prod_assoc_receipt` dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

- $14,887$ ($< 0.005\%$) of these tickets contain "sushi rice"
- 431 ($< 0,00015\%$)
  contain "nori seaweed, wasabi, sushi rice, rice vinegar"

An example on retail data

Our `prod_assoc_receipt` dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

- $14,887$ ($< 0.005\%$) of these tickets contain "sushi rice"
- 431 ($< 0,00015\%$)
  contain "nori seaweed, wasabi, sushi rice, rice vinegar"
- 133 ($< 0.00004\%$)
  contain "nori seaweed, wasabi, sushi rice, soy sauce"

An example on retail data

Our `prod_assoc_receipt` dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

- $14,887$ ($< 0.005\%$) of these tickets contain "sushi rice"
- 431 ($< 0,00015\%$)
  contain "nori seaweed, wasabi, sushi rice, rice vinegar"
- 133 ($< 0.00004\%$)
  contain "nori seaweed, wasabi, sushi rice, soy sauce"

Requires respectively $k = 23$ and $k = 255$.

Inputs

- Customer base
  Demographic groups as attributes - *age range, gender, region*
- Products taxonomy
  cat(chocolate cream) = {*Fresh food, Dairy, Ultra fresh, Desserts*}
- Tickets
  Centralized nation-wide every night

\+ a set of *targets*: categories or products to be studied.

Dealing with a large results set

With proper preparation and constrains

- mining association rules is feasible in a batch

But

- A target product/category yields 100 to 10,000 associations
- Casual analysis concerns dozens targets
- Analyst have the time for 10 “top” rules (or at most 20)

Dealing with a large results set

With proper preparation and constrains

- mining association rules is feasible in a batch

But

- A target product/category yields 100 to 10,000 associations
- Casual analysis concerns dozens targets
- Analyst have the time for 10 "top" rules (or at most 20)

We need to sort rules

Sorting association rules

Existing work provides more than 39 quality measures [geng2006ACM,Lenca2007].

- Which one should we pick ?
- Are prior studies relevant to tickets analysis?
- In some domains reference rules can be found [LeSANER15]

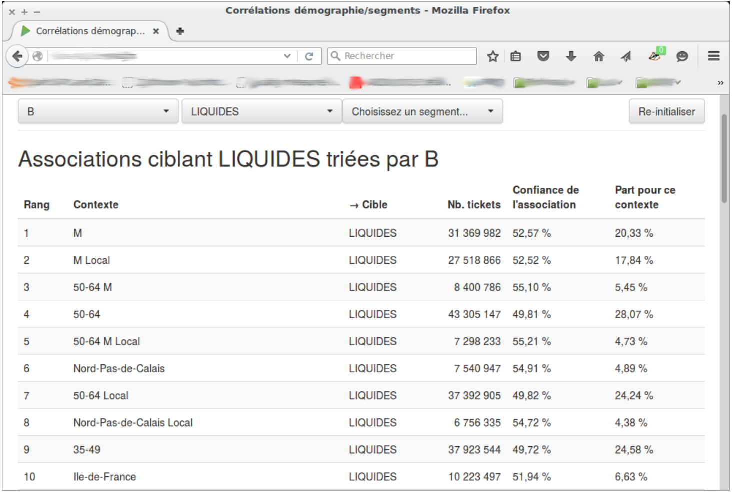
TODO refs complètes

CAPA: Comparative Analysis of PAtterns

We present CAPA , the system allowing us to answer:

1. In terms of ranking, how different are interestingness measures?
2. Which ones are meaningful to marketing analysts at this scale?

The exploration application



Corrélations démographie/segments - Mozilla Firefox

Corrélations démograp...

Rechercher

B | LIQUIDES | Choisissez un segment... | Re-initialiser

Associations ciblant LIQUIDES triées par B

Rang	Contexte	→ Cible	Nb. tickets	Confiance de l'association	Part pour ce contexte
1	M	LIQUIDES	31 369 982	52,57 %	20,33 %
2	M Local	LIQUIDES	27 518 866	52,52 %	17,84 %
3	50-64 M	LIQUIDES	8 400 786	55,10 %	5,45 %
4	50-64	LIQUIDES	43 305 147	49,81 %	28,07 %
5	50-64 M Local	LIQUIDES	7 298 233	55,21 %	4,73 %
6	Nord-Pas-de-Calais	LIQUIDES	7 540 947	54,91 %	4,89 %
7	50-64 Local	LIQUIDES	37 392 905	49,82 %	24,24 %
8	Nord-Pas-de-Calais Local	LIQUIDES	6 756 335	54,72 %	4,38 %
9	35-49	LIQUIDES	37 923 544	49,72 %	24,58 %
10	Ile-de-France	LIQUIDES	10 223 497	51,94 %	6,63 %

Interestingness measures

- We have rules as $A \rightarrow b$, where b is a target category/product
- We know how many transactions contain A, b, and $A \cup \{b\}$

Interestingness measures

- We have rules as $A \rightarrow b$, where b is a target category/product
- We know how many transactions contain A, b, and $A \cup \{b\}$

We select **39 interestingness measures** relying on these numbers only (See Table 4)
How do the resulting lists compare?

Evaluation method

We let experts judge which top-results are interesting.
But 39 rankings is too much.

Evaluation method

We let experts judge which top-results are interesting.
But 39 rankings is too much.

We evaluate in two steps:

1. Automatic distinction of 5 families of measures, which are ranking similarly
   denoted $G_{\{1,2,3,4,5\}}$
2. The user study itself, where we compare those 5 families.

Comparing rankings

In each scenario, we generate rules, rank them, and their 39 scores.

We do pairwise comparisons of the sorted lists using:

- Spearman's rank correlation coefficient
- Kendall's τ
- Overlap@20
- NDCC: Normalized Discounted Correlation Coefficient
  our adaptation of NDCG[?]

Comparing rankings

In each scenario, we generate rules, rank them, and their 39 scores.

We do pairwise comparisons of the sorted lists using:

- Spearman's rank correlation coefficient
- Kendall's τ
- Overlap@20
- NDCC: Normalized Discounted Correlation Coefficient
  our adaptation of NDCG[?]

Similarity coefficients are averages over 64 targets

Clustering

Apply average linkage over the averaged correlation matrix.

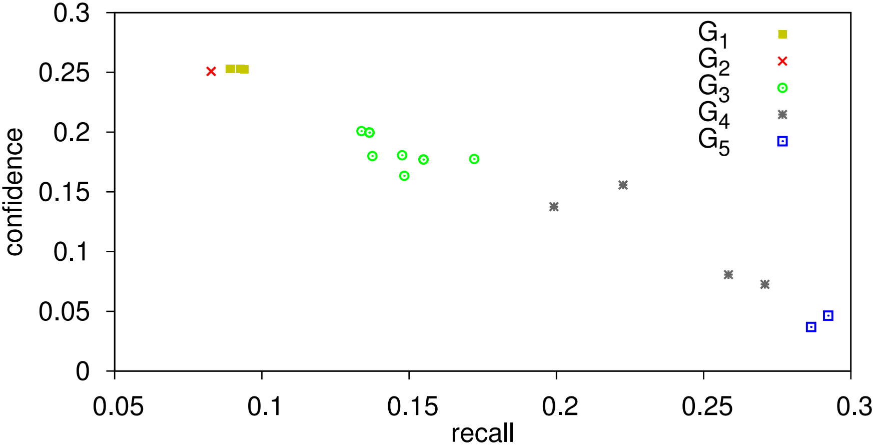
Clustering

Apply average linkage over the averaged correlation matrix.

Yields 5 clusters, $G_{\{1,2,3,4,5\}}$

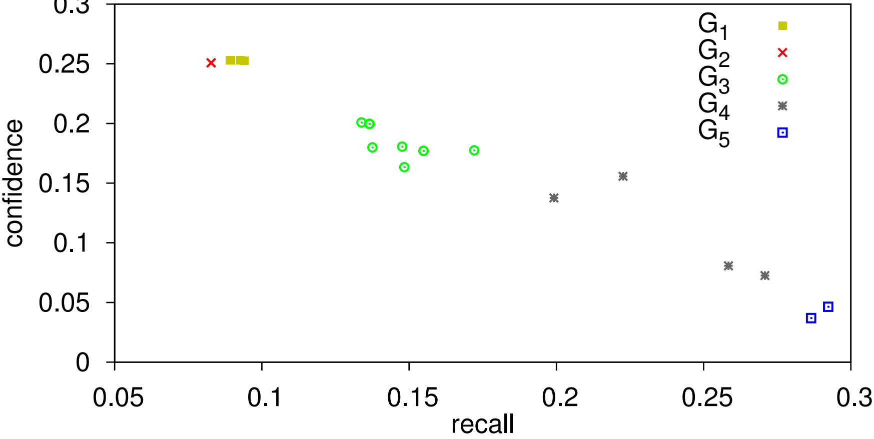
- Many measures rank as *confidence* (in G_1)
- Families consistent over `demo_assoc` , `prod_assoc_client` and `prod_assoc_receipt`
- Some rankings are even equal
- Ranking by *p-value* is similar to simpler measures.

Measures families' behavior



Average recall/confidence of the top-20 results of each measure

Measures families' behavior



Average recall/confidence of the top-20 results of each measure

Which one are more meaningful to marketing experts?

User study

2 marketing experts from Intermaché are given a free access to results from `demo_assoc` , `prod_assoc_client` and `prod_assoc_receipt` .

Rules can be ranked according to a representative measure of each of the 5 clusters - presented as A, B, C, D, E.

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?
  Quality of the top results do vary with the ranking
  Sometimes rankings are equally interesting

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?
  Quality of the top results do vary with the ranking
  Sometimes rankings are equally interesting
- How many rules are interesting?

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?
  Quality of the top results do vary with the ranking
  Sometimes rankings are equally interesting
- How many rules are interesting?
  10, at most 20

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?
  Quality of the top results do vary with the ranking
  Sometimes rankings are equally interesting
- How many rules are interesting?
  10, at most 20
- Do those rules overlap with your experience or former studies?

User study

Experts are asked for their favorite(s) measure(s) in each scenario.

Other questions from their interview:

- What is your first impression?
  Quality of the top results do vary with the ranking
  Sometimes rankings are equally interesting
- How many rules are interesting?
  10, at most 20
- Do those rules overlap with your experience or former studies?
  $\rightarrow$ their first evaluation criterion

User study: on demographic rules

- Preference towards families G_1 and G_3
- Expect finding *both* extremes of associations
  - Strong ones: *{50-64}* $\rightarrow$ *pet food*
  - Weak ones: *{<35, Paris}* $\rightarrow$ *pet food*

User study: on products associations

Additional option: associate only with products from the same category.

User study: on products associations

Additional option: associate only with products from the same category.

- Without filter: favor G_1 and G_2, *ie.* high confidence associations
- When filtering: favor G_3 and G_4, *ie.* balanced rankings

User study: on products associations

Additional option: associate only with products from the same category.

- Without filter: favor G_1 and G_2, *ie.* high confidence associations
- When filtering: favor G_3 and G_4, *ie.* balanced rankings

The key factor: differenciating
{vanilla cream, emmental}$\rightarrow$ *chocolate cream* (32% confidence)
with
{vanilla cream}$\rightarrow$ *chocolate cream* (31% confidence)

CAPA 's main results

- In terms of ranking, quality measures may be surprisingly similar
- Simple computations (Confidence, Piatetsky-Shapiro's) yield good satisfaction

CAPA 's main results

- In terms of ranking, quality measures may be surprisingly similar
- Simple computations (Confidence, Piatetsky-Shapiro's) yield good satisfaction

Possible evolutions

- Switching the system to in-memory storage
- Presentation enhancements of ranked rules

Perspectives

- Going distributed

Perspectives

- Going distributed
  - MapReduce version of TopPI currently under review

Perspectives

- Going distributed
  - MapReduce version of TopPI currently under review
- Re-ranking each *top*(i)

cf. *Testing Interestingness Measures in Practice: A Large-Scale Analysis of Buying Patterns*, Kirchgessner, Leroy, Amer-Yahia, Mishra @ DSAA'16

Going distributed

CIS enumeration is slower with long transactions ($>$ 1000 items)

The WebDocs dataset, with $k = 10$

TopPI takes almost 10 hours.
The baseline only fills 3% of $top(i)$ in a day.

http://fimi.ua.ac.be/data/webdocs.dat.gz

Going distributed

CIS enumeration is slower with long transactions (> 1000 items)

The WebDocs dataset, with $k = 10$

TopPI takes almost 10 hours.
The baseline only fills 3% of $top(i)$ in a day.

We created a MapReduce version of TopPI

- currently under review

http://fimi.ua.ac.be/data/webdocs.dat.gz

Which value is relevant for k?

Back to the most frequent CIS containing sushi rice:

Rank	Support	Itemset
1	$14,887$	"sushi rice"
...		...
24	431	"sushi rice, nori seaweed, wasabi, rice vinegar"
...		...
255	133	"sushi rice, nori seaweed, wasabi, soy sauce"

Re-ranking

Itemsets in each $top(i)$ are sorted by decreasing frequency.

- each $top(i)$ can be re-ordered with a finer quality measure
- keep only the $n < k$ results

Re-ranking

Itemsets in each $top(i)$ are sorted by decreasing frequency.

- each $top(i)$ can be re-ordered with a finer quality measure
- keep only the $n < k$ results
- typically $k \in [100; 500]$ and $n \in [10; 50]$
- may require an additional pass on the dataset

cf. *Testing Interestingness Measures in Practice: A Large-Scale Analysis of Buying Patterns*, Kirchgessner, Leroy, Amer-Yahia, Mishra @ DSAA'16

Item-Centric Mining in a nutshell

Return, for each item, its k most frequent closed itemsets.

- intuitive parameter, k
- intuitive results organization, per item.

The TopPI algorithm

- efficiently computes all top-k lists at once
- scales from a laptop to a high-end server
- robust from 1 to 300 million transactions

Source code (including Hadoop version) available at
`https://github.com/slide-lig/TopPI`