# TopPI
# An Efficient Algorithm for Item-Centric Mining

**Martin Kirchgessner**[1] Vincent Leroy[1] Alexandre Termier[2]
Sihem Amer-Yahia[1] Marie-Christine Rousset[1]

Laboratoire d'Informatique de Grenoble
[1]firstname.lastname@imag.fr
[2]firstname.lastname@irisa.fr

DaWaK, Porto - September 6, 2016

Item-Centric Mining?

# An example on retail data

Our *Tickets* dataset represents 290 million receipts from 1800 french supermarkets.

Which sets of products frequently include sushi rice?

# An example on retail data

Our *Tickets* 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 *Tickets* 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 *Tickets* 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 *Tickets* 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"

Then, which sets include rice vinegar? soy sauce? ...

# An example on retail data

Our *Tickets* 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"

Then, which sets include rice vinegar? soy sauce? ...

**Item-Centric Mining**

Mining a collection of itemsets providing a few itemsets about any item.

# Transactional datasets

## Input

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

# 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

# Big transactional datasets

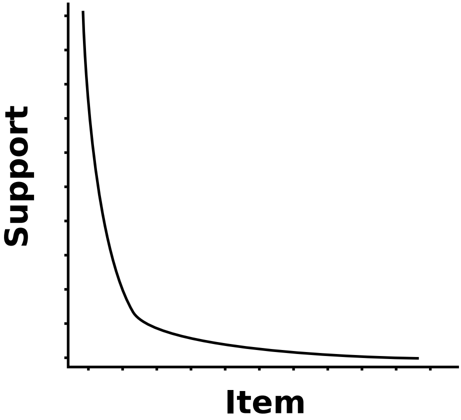
"big" means our datasets contain at least

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

# Big transactional datasets

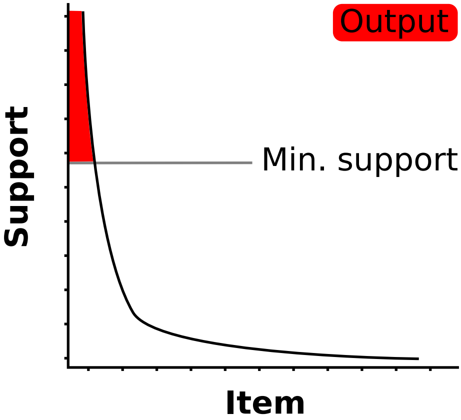
"big" means our datasets contain at least

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



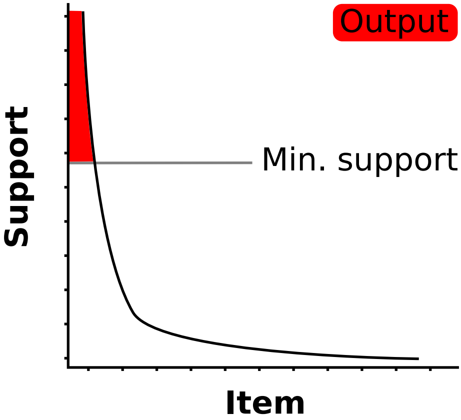

[2] *The Long Tail: Why the Future of Business Is Selling Less of More*, Anderson (2006)

# Frequent Itemset Mining on big datasets



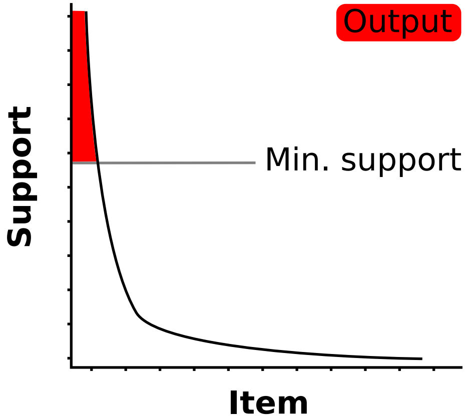

- Which minimum support yields interesting results?

# Frequent Itemset Mining on big datasets



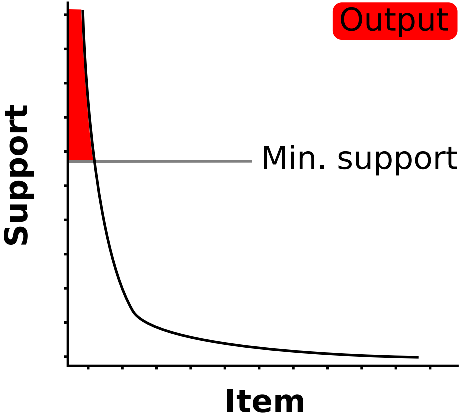

- Which minimum support yields interesting results?
- Are all closed itemsets interesting?

# Frequent Itemset Mining on big datasets

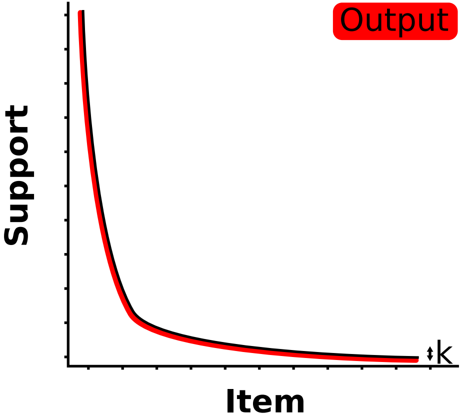



- Which minimum support yields interesting results?
- Are all closed itemsets interesting?
- What about the remaining items?

# 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

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

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

# 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...

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 all heaps $top(i)$.
2. Progressively fill them by enumerating CIS...

# TopPI 's main program

1. Instantiate all heaps $top(i)$.
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 all heaps $top(i)$.
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

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 $\geq 100$?
  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 $\geq 100$?
  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 $\{b, c, d\}$.

# 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**

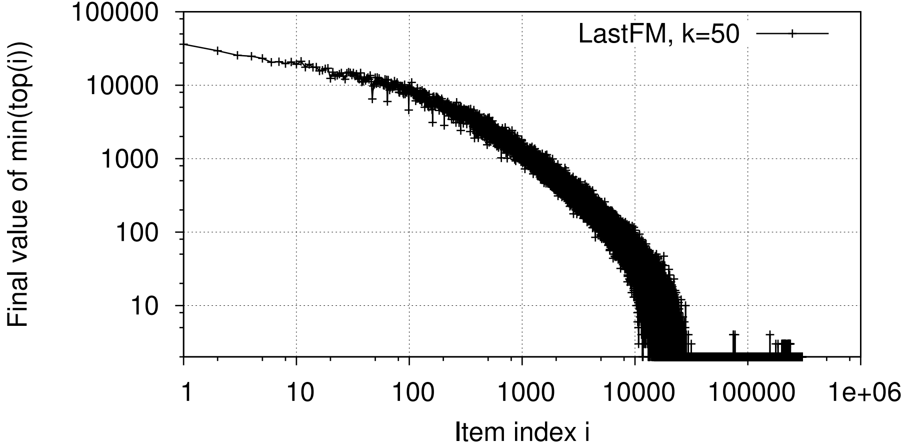
If, $\forall i < max(P), min(top(i)) \geq support_{\mathcal{D}}(P)$,
then the branch rooted at $P$ can be pruned.

# Deciding quickly to prune with prefix short-cutting

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

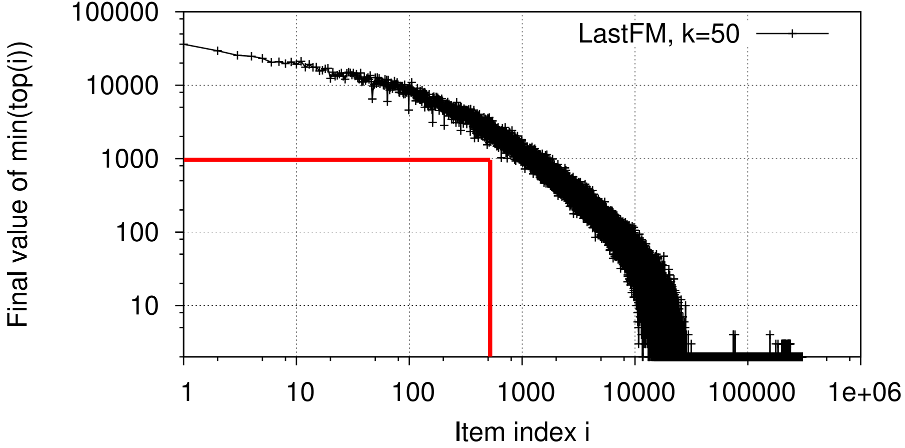
# Deciding quickly to prune with prefix short-cutting

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



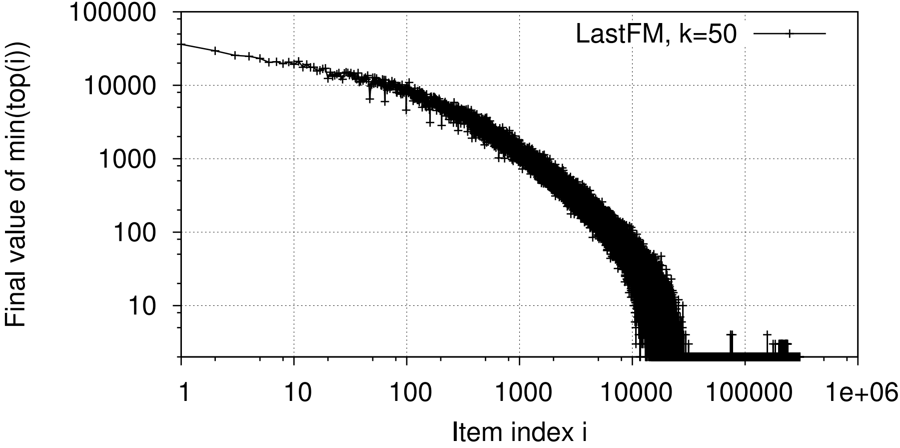
# Deciding quickly to prune with prefix short-cutting

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

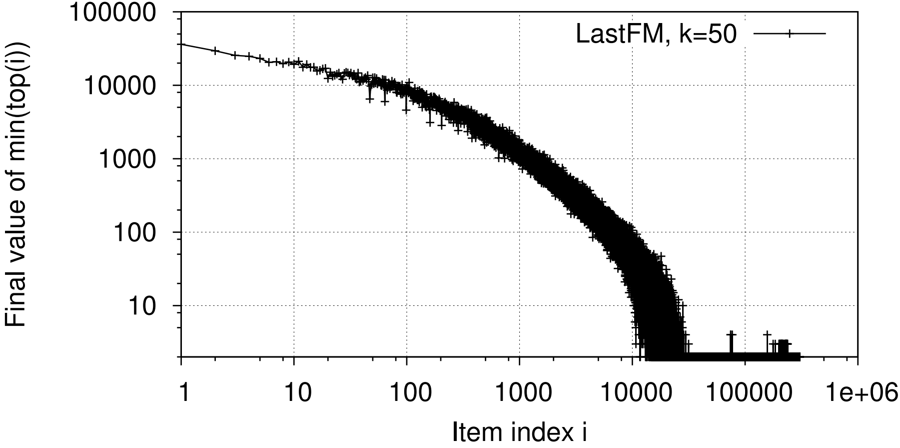



Here if $support_{\mathcal{D}}(P) \leq 1000$, no need to test $min(top(i))$ for $i < 500$.

# Dynamic threshold adjustment



# Dynamic threshold adjustment




**Dynamic threshold adjustment**

Finding a minimum frequency threshold adapted to each CIS branch.

# 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** | $\vert\mathcal{I}\vert$ | $\vert\mathcal{D}\vert$ | **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

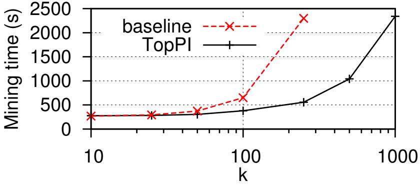
| Dataset | $\vert\mathcal{I}\vert$ | $\vert\mathcal{D}\vert$ | File size |
|---|---|---|---|
| *Tickets* | $222,228$ | $290,734,163$ | 24GB |
| *Clients* | $222,228$ | $9,267,961$ | 13.3GB |
| *LastFM* | $1,206,195$ | $1,218,831$ | 277MB |

## We measure run-times

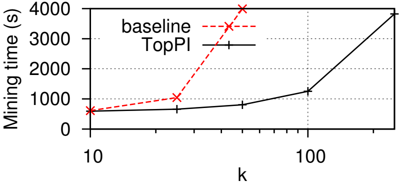
- Averaged over 3 attempts
- Not including the time to load $\mathcal{D}$.
- On a single server:
  - 2 Intel Xeon E5-2650, providing 16 cores with Hyper Threading
  - 128GB of RAM

All programs are implemented in Java.

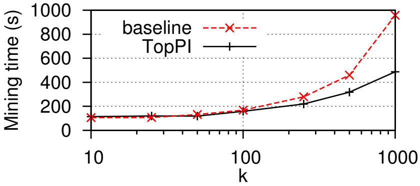
# TopPI and Baseline run-times




*Tickets*




*Clients*




*LastFM*

(using 16 threads)

# Contributions Impact

| Dataset | TopPI |
|---|---|
| *Tickets* | 222 s. |
| *Clients* | 661 s. |
| *LastFM* | 116 s. |

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

# Contributions Impact

| Dataset | TopPI | Without 3.5 |
|---|---|---|
| *Tickets* | 222 s. | 1136 ($\times 5$) |
| *Clients* | 661 s. | Out of mem. |
| *LastFM* | 116 s. | 177 (+53%) |

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

Section 3.5: Dynamic threshold adjustment

# Contributions Impact

| Dataset | TopPI | Without 3.5 | Without 3.6 |
|---|---|---|---|
| *Tickets* | 222 s. | 1136 ($\times$5) | 230 (+4%) |
| *Clients* | 661 s. | Out of mem. | 4177 ($\times$6) |
| *LastFM* | 116 s. | 177 (+53%) | 150 (+29%) |

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

Section 3.5: Dynamic threshold adjustment
Section 3.6: Pruning with prefix short-cutting

# Contributions Impact

| Dataset | TopPI | Without 3.5 | Without 3.6 | Without both |
|---|---|---|---|---|
| *Tickets* | 222 s. | 1136 ($\times$5) | 230 (+4%) | 3.8 hours, $\times$62 |
| *Clients* | 661 s. | Out of mem. | 4177 ($\times$6) | Out of memory |
| *LastFM* | 116 s. | 177 (+53%) | 150 (+29%) | 243 ($\times$2) |

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

Section 3.5: Dynamic threshold adjustment
Section 3.6: Pruning with prefix short-cutting

# 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

# Item-Centric Mining in a nutshell

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

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

# 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

Thank you for your attention.

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