TopPI
An Efficient Algorithm for Item-Centric Mining

Martin Kirchgessner\textsuperscript{1}  Vincent Leroy\textsuperscript{1}  Alexandre Termier\textsuperscript{2}
Sihem Amer-Yahia\textsuperscript{1}  Marie-Christine Rousset\textsuperscript{1}

Laboratoire d’Informatique de Grenoble
\textsuperscript{1}firstname.lastname@imag.fr
\textsuperscript{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?

14,887 (< 0.005%) 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”
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 $t_j \subseteq \mathcal{I}$. 

Output (presented to the analyst)

A collection of closed itemsets (CIS), i.e., itemsets $P$ satisfying $\forall Q \supset P \text{ s.t. } \text{support}_{\mathcal{D}}(P) = \text{support}_{\mathcal{D}}(Q)$.

Where $\text{support}_{\mathcal{D}}(P) = |\{ t \in \mathcal{D} | P \subset t \}|$.
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 $t_j \subseteq \mathcal{I}$.

Output (presented to the analyst)
A collection of closed itemsets (CIS),
ie. itemsets $P$ satisfying $\nexists Q \supset P$ s.t. $\text{support}_\mathcal{D}(P) = \text{support}_\mathcal{D}(Q)$.

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

[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 $I$
- Millions of transactions in $D$
Big transactional datasets

“big” means our datasets contain at least

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

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

Output

Support

Min. support

Item
Replace the minimum support by a single parameter, $k$
TopPI ’s problem statement

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

TopPI stands for “Top Per Item”.

---

M. Kirchgessner (LIG)  TopPI : Item-Centric Mining  DaWaK’16  8 / 25
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 Per Item”.

Benefits

- Restrict intuitively the CIS space
Item-Centric Mining

**TopPI’s problem statement**

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

**Benefits**

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

TopPI stands for “Top Per Item”.

M. Kirchgessner (LIG)
Item-Centric Mining

**TopPI’s problem statement**

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

TopPI stands for “Top Per Item”.

**Benefits**

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

We target high-end, multi-core servers.
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 $\text{top}(i)$.

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

Easy to parallelize, fine for small files.

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 \(\text{top}(i)\).

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 \( i \).

Li, Wang, Zhang, Zhang, Chang @ RecSys’08
PFP: parallel FP-Growth

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

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 \( \text{top}(i) \) for an item \( i \).
Efficiently enumerating CIS

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

Enumeration is inspired from PLCM.

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 $\text{top}(i)$ 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 $\text{max}(P)$.


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 $\text{top}(i)$.
2. Progressively fill them by enumerating CIS... and prune the enumeration when the concerned items already have a complete $\text{top}(i)$.

We can poll each item’s heap via $\text{min}(\text{top}(i))$: the smallest itemset support in $\text{top}(i)$. 
An example

After enumerating \( \{c, d\} (support = 100) \) → 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 \( \text{top}(c) \) and \( \text{top}(d) \).

Then, before attempting to find \( \{b, c, d\} \)
- we know that \( \text{support}_D(\{b, c, d\}) \leq 100 \)
- Can we prune if \( \text{top}(b) \), \( \text{top}(c) \) and \( \text{top}(d) \) already have \( k \) CIS of support \( \geq 100 \)?
  ie. \( \text{min}(\text{top}(b)) \geq 100 \), idem for \( c \) and \( d \).
An example

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

Then, before attempting to find \( \{b, c, d\} \)
  - we know that \( \text{support}_D(\{b, c, d\}) \leq 100 \)
  - Can we prune if top(b), top(c) and top(d) already have \( k \) CIS of support \( \geq 100 \)?
    ie. \( \text{min}(\text{top}(b)) \geq 100 \), 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 \( \text{min}(\text{top}(a)) \leq 100 \), we cannot prune \( \{b, c, d\} \).
Pruning in TopPI

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

- $\max(Q) = \max(P)$
- $\text{support}_D(Q) \leq \text{support}_D(P)$

**TopPI’s basic pruning principle**

If, $\forall i < \max(P), \min(\text{top}(i)) \geq \text{support}_D(P)$, 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 $min(top(i))$, $\forall i < max(P)$, $\forall P$.

![Graph showing the final value of $min(top(i))$ vs. item index $i$. The graph is labeled with "LastFM, k=50." The x-axis represents item index $i$ ranging from 1 to $1e+06$, and the y-axis represents the final value of $min(top(i))$ ranging from 1 to 1,000,000.]
Deciding quickly to prune with prefix short-cutting

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

Here if \( \text{support}_D(P) \leq 1000 \), no need to test \( \min(top(i)) \) for \( i < 500 \).
Dynamic threshold adjustment

Finding a minimum frequency threshold adapted to each CIS branch.

M. Kirchgessner (LIG)

TopPI : Item-Centric Mining

LastFM, k=50
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   | |I|   | |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

<table>
<thead>
<tr>
<th>Dataset</th>
<th></th>
<th></th>
<th>File size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tickets</td>
<td>222, 228</td>
<td>290, 734, 163</td>
<td>24GB</td>
</tr>
<tr>
<td>Clients</td>
<td>222, 228</td>
<td>9, 267, 961</td>
<td>13.3GB</td>
</tr>
<tr>
<td>LastFM</td>
<td>1, 206, 195</td>
<td>1, 218, 831</td>
<td>277MB</td>
</tr>
</tbody>
</table>

We measure run-times

- Averaged over 3 attempts
- Not including the time to load $D$
- 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**

- **TopPI**
- **Baseline**

**Clients**

(Using 16 threads)

**LastFM**

- **TopPI**
- **Baseline**
**Contributions Impact**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TopPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tickets</td>
<td>222 s.</td>
</tr>
<tr>
<td>Clients</td>
<td>661 s.</td>
</tr>
<tr>
<td>LastFM</td>
<td>116 s.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TopPI</th>
<th>Without 3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tickets</td>
<td>222 s.</td>
<td>1136 (×5)</td>
</tr>
<tr>
<td>Clients</td>
<td>661 s.</td>
<td>Out of mem.</td>
</tr>
<tr>
<td>LastFM</td>
<td>116 s.</td>
<td>177 (+53%)</td>
</tr>
</tbody>
</table>

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

Section 3.5: Dynamic threshold adjustment
### Contributions Impact

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TopPI</th>
<th>Without 3.5</th>
<th>Without 3.6</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Tickets</em></td>
<td>222 s.</td>
<td>1136 (×5)</td>
<td>230 (+4%)</td>
</tr>
<tr>
<td><em>Clients</em></td>
<td>661 s.</td>
<td>Out of mem.</td>
<td>4177 (×6)</td>
</tr>
<tr>
<td><em>LastFM</em></td>
<td>116 s.</td>
<td>177 (+53%)</td>
<td>150 (+29%)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TopPI</th>
<th>Without 3.5</th>
<th>Without 3.6</th>
<th>Without both</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tickets</strong></td>
<td>222 s.</td>
<td>1136 (×5)</td>
<td>230 (+4%)</td>
<td>3.8 hours, ×62</td>
</tr>
<tr>
<td><strong>Clients</strong></td>
<td>661 s.</td>
<td>Out of mem.</td>
<td>4177 (×6)</td>
<td>Out of memory</td>
</tr>
<tr>
<td><strong>LastFM</strong></td>
<td>116 s.</td>
<td>177 (+53%)</td>
<td>150 (+29%)</td>
<td>243 (×2)</td>
</tr>
</tbody>
</table>

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

Testing Interesting Measures in Practice: A Large-Scale Analysis of Buying Patterns, Kirchgessner, Leroy, Amer-Yahia, Mishra @ DSAA’16
Perspectives

- Going distributed
  - MapReduce version of TopPI currently under review
- Re-ranking each \( \text{top}(i) \)

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