Computing Frequent $k$-Itemsets Directly in Sparse Datasets

M. Atzori$^{1,2}$  P. Mancarella$^1$  F. Turini$^1$

$^1$Department of Computer Science
University of Pisa

$^2$Information Science and Technology Institute
CNR, Pisa

Speaker: Maurizio Atzori
atzori@di.unipi.it

Fourth International Workshop on Knowledge Discovery in Inductive Databases (KDID) 2005
1 Motivation
   • The Problem of Mining Frequent Itemset
   • Some Known Solutions to Reduce Memory Requirements

2 Our Results/Contribution
   • The Basic Idea of Our Proposal
   • Results
Frequent (k-)Itemset Mining is Useful

Notations

- Frequent itemsets are used to compute
  - Association analysis
  - Rule based classification
  - Clustering

Equation (Not so difficult...)

Frequent itemsets = frequent k-large itemsets for every k

- We will focus on $\sigma$-frequent $k$-itemset mining
  (from a dataset $\mathcal{D}$ over the set of items $\mathcal{I}$)
  - $k$-itemset – itemset of size $k$
  - $\sigma$-frequent – the itemset appears in at least $\sigma\%$ of $\mathcal{D}$
Frequent Itemset Mining is Memory Consuming
There is a trade-off memory usage and number of passes

(Very Usual) Assumptions

1. \( \mathcal{I} \) can fit into main memory, \( \mathcal{I} \times \mathcal{I} \) can’t.
2. \( \mathcal{D} \) can’t, neither.

- Using levelwise approaches
  - \( O(k) \) passes through the dataset (Good)
  - Candidate itemsets of level \( k \) can be \( \binom{|\mathcal{I}|}{k} \in O(|\mathcal{I}|^k) \) (Bad!)

- Using depth-first approaches
  - few (constant) passes through the dataset (Very Good)
  - data structures require \( O(|\mathcal{D}|) \) space (Extremely bad!)

- The output size and the memory requirements grow fast by decreasing \( \sigma \)
Possible Solutions to Fit into Memory

- Hashing itemset counts (in a levelwise approach)
  - compute actual counts using an hashtable smaller than the set of candidates, and then prune according to the counts
  - no guarantee to work, especially if many *candidates occur in the dataset*

- Partitioning (in both approaches)
  - we can have a huge number of (hopefully small) sets of candidates
  - if the small sets are not very similar (i.e., if the dataset is *not very uniform*) it doesn’t work

- A very simple one, effective (levelwise approach)
  - generate candidate itemsets of level $k$
  - compute the count of such candidates *in several passes*, by fitting into memory only a small subset each time
Basic Idea: $\mathcal{D}$ can be transformed into a stream of $k$-itemsets.

Example

\[ \mathcal{D} = \{\{a, b, d\}, \{a, c, e\}, \{a, d, f\}, \{b, c\}, \{b, d, e\}, \{c, d, f\}\} \]

\[ s_1 = \langle\{a, b\}, \{a, d\}, \{b, d\}\rangle \]
\[ s_2 = \langle\{a, c\}, \{a, e\}, \{c, e\}\rangle \]
\[ s_3 = \langle\{a, d\}, \{a, f\}, \{d, f\}\rangle \]
\[ s_4 = \langle\{b, c\}\rangle \]
\[ s_5 = \langle\{b, d\}, \{b, e\}, \{d, e\}\rangle \]
\[ s_6 = \langle\{c, d\}, \{c, f\}, \{d, f\}\rangle \]

\[ S_\mathcal{D} = s_1 :: s_2 :: s_3 :: s_4 :: s_5 :: s_6 \]
Memory and Number of Passes Required.

- We developed an algorithm for frequent $k$-itemset mining by exploiting an existing Iceberg Queries Algorithm.
- Space complexity $O\left(\frac{(m_D^k)}{\sigma}\right)$
  - it does not depend on $|\mathcal{D}|$ (Good!)
  - it does not depend on $|\mathcal{I}|$ (Good!)
  - it depends on $m_D$, the longest transaction in $\mathcal{D}$ (Good, if $\mathcal{D}$ is sparse enough)
- Only 2 passes through the dataset (3, if we don’t know $m_D$ in advance)
Experiments.

- By replicating (with slight changes in each transaction) RETAIL we obtained a dataset with 12 millions of transactions and 16470 different items.
- We truncated such $\mathcal{D}$ at 1, 2, 3, ... millions of transactions and computed frequent 2-itemset ($\sigma = 0.01 = 1\%$):
  - Relim computed frequent itemset up to 3 millions, then crashed
  - Apriori, FP-Growth and Eclat worked up to 4 millions
  - Crashes were due to insufficient memory (512Mb Ram used)
  - Our algorithm used a constant amount of memory and scaled up linearly (in time)
  - Our algorithm never crashed
Summary

- Frequent \((k-)\)itemset mining can be very memory consuming, unless performing several passes through the dataset.
- For sparse datasets, the algorithm we developed is extremely memory saving for computing frequent \(k\)-itemsets;
- Memory requirement depends only on \(\sigma\) and \(k\), and the number of passes is constant (2 or 3).

Future Work
- Optimized implementation.
- a hybrid version with a second level-wise step.
B. Goethals. 
Memory Issues In Frequent Itemset Mining.  
*Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*, Nicosia, Cyprus, March 14-17, 2004

G. Grahne and J. Zhu.  
Mining Frequent Itemsets from Secondary Memory.  
*Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, 1-4 November 2004, Brighton, UK