Fast Algorithms for Mining Association Rules
Rakesh Agrawal, Ramakrishnan Srikant

Presented by Wenhao Xu
Discussion led by Sophia Liang

Outline
- This is an important paper because
  - VLDB 10 Years Best Paper Award
  - Has been 1st highest cited paper of all papers in the fields of databases and data mining until 2007 in Citeseer
  - 2009 Citeseer Citations: Rank 18 in all computer science papers
  - Two authors: all better jobs!!!

Why it is so important?
- It addresses an important problem.
- It proposes an algorithm that is better than previous algorithms
- Lots of papers afterwards are based on its basic concepts

Agenda

Example of Association Rule Mining

Amazon

For Amazon: Earn more money!
For you: Good user experience!

Example & Notions

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Items</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{milk, diaper, beer, Coke}</td>
<td>(milk, diaper) ➝ (beer)</td>
</tr>
<tr>
<td>2</td>
<td>{milk, bread}</td>
<td>(milk) ➝ (bread)</td>
</tr>
<tr>
<td>3</td>
<td>{milk, bread, diaper, beer}</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{milk, bread, diaper, coke}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{bread, diaper, beer, eggs}</td>
<td></td>
</tr>
</tbody>
</table>

- **Item Sets**: a set of items, like {milk, diaper}, is an item set.
- **Association rule**: implication in the form of X ➝ Y; X and Y are both item sets.
- **Implication** means co-occurrence, not causation
- **Support of the rule**: the fraction of transactions that contain both X and Y. i.e. \( F(X, Y) \)
- \( F(X, Y) = F({milk, diaper, beer}) = 2/5 \)
- **Confidence of the rule**: the ratio of transactions that contain X contain Y, i.e. \( C(X, Y) = F(X \cup Y) / F(X) \)
- \( C({milk, diaper}) → (beer) = F({milk, diaper, beer}) / F({milk, diaper}) = \) (2/5)/(3/5) = 2/3

Formal definition: Association Rule Mining

- Given a large set of transaction D, generate all association rules that have support and confidence greater than the user-specified minimum support (called **minsup**), and minimum confidence (called **minconf**) respectively.
- **minsup & minconf**: ensure usefulness
- **Large**:
  - A significant of data sets in data mining
  - require effective algorithms

Generic Algorithms

- **Step 1**: Find all itemsets that have transaction support above minimum support. These itemsets are called **large itemsets**.
  - Focus of this paper: find large itemsets
    - AIS, SETM
    - Aprior, ApriorTid, ApriorHybrid
- **Step 2**: Use the large itemsets to generate the desired rules.
  - A straightforward algorithm:
    For every large itemset I:
      - For every non-empty subset a of I,
        - rule: a ➝ (L-a)
        - \( C(rule) >= minconf \)
      - output
      - endfor
    - endfor
- Refer to <fast algorithms for mining association rules in large databases> for a fast algorithm
Apriori: Find Large Itemsets

• Basic Concepts:
  – Generate all possible candidate large itemsets: Any subset of a large itemset must be large
  – Filter out small itemsets

• Assumption: items with in an itemset are kept in lexicographic order

• Basic steps:
  – Generate candidate k-itemsets from large (k-1)-itemsets
  – For each candidate k-itemsets, calculate its support;
  – If its support is larger than minsup, add it to large k-itemsets
  – Continue the above three steps by adding 1 to k until no large k-itemsets are found.

Apriori-gen

• Join step
  insert into C_k select p.item_1, p.item_2, ..., p.item_k-1, q.item_k-1
  from L_{k-1} p, L_{k-1} q
  where p.item_1 = q.item_1, ..., p.item_k-2 = q.item_k-2,
  p.item_{k-1} < q.item_k-1

• Prune step
  delete all itemsets that have some (k-1)-subsets which are not in (k-1) large itemsets

Example

L_3 = \{ (1 2 3), (1 2 4), (1 3 4), (1 3 5), (2 3 4) \}

After joining

\{ (1 2 3 4), (1 3 4 5) \}

After pruning

\{ 1 2 3 4 \}

AprioriTid & AprioriHybrid

• Still use apriori-gen to generate candidate itemsets
• Try to reduce the times of scan the database
  – Use a candidate set (called D_k) that include TID of the corresponding transaction
  – Due to the time limit, refer to the paper by yourself
• D_k could be smaller than the whole database when k is large and can fit into the memory
• However, may be slower than Apriori because when k is small, D_k is even larger than the original transaction.
• AprioriHybrid to combine the benefits of Apriori and AprioriTid by using a heuristic to switch from Apriori to AprioriTid on the fly.

AprioriTid

Scan the whole database every time

Evaluation

• Use 6 sets of the synthetic data
• An IBM RS/6000 530H workstation
• Compare to SETM and AIS
Recent Development

• FP-tree
  – Refer to “Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach”, Jiawei Han, Jian Pei, et al.
  – About an order of magnitude faster than Aprori
• I don’t find other significant improved approaches.

Conclusion

• Important problem
• Good paper as a foundation of association-rule mining
• Can be improved
  – FP-tree

Thanks for your attention!