MINING FREQUENT GENERALIZED PATTERNS FOR WEB PERSONALIZATION

P. GIANNIKOPOULOS, University of Peloponnese
I. VARLAMIS, Athens University of Economics and Business
M. EIRINAKI, San Jose State University

Mining Social Data - ECAI 2008
Motivation

- Huge amounts of social data generated in a regular basis
- Information overload remains a problem, even if the user chooses to monitor a subset of those (e.g. using blog aggregators)
- Need to personalize the content by providing recommendations to the user
Motivation

• Association Rule Mining (ARM): data mining technique used to identify item co-occurrence in “market baskets”
  – E.g. items bought, web pages visited, etc.
  – $A, B \rightarrow C$ [confidence, support]

• Useful for making recommendations

• PROBLEM: In environments with continually updated content (blogs, newspaper sites), we have limited or no transaction information for new items
  – The **cold-start problem**: how to make suggestions in the absence of transaction history (i.e. for the new items)
Motivation

• SOLUTION: Generalize frequent patterns using hierarchical organization of the content (i.e. taxonomies)

  – Generalized Pattern Mining algorithms [AprioriAll, AprioriSome, DynamicSome, CDIST, CDIST_O, CWIN, CMINWIN, Collaborative Filtering, Content-Based Filtering, Markov Models, Sequential Pattern Mining algorithm, GSP, FP-Growth, GP-Close]
Our Solution

• Combine the forces of:
  – *FP-Growth*: An association rule mining algorithm
  – *GP-Close*: A generalized pattern mining algorithm

• Why both?
  – When the performance of the FP-Growth algorithm is combined with the taxonomic features of GP-Close we are able to provide a solution to the cold-start problem
Outline

• Motivation
• Preliminaries
  – FP-Growth
  – GP-Close
• The FGP Algorithm
• Experimental Evaluation
• Future Work
• References
FP-Growth (1/2)

• A well-known association rule mining (ARM) algorithm, proposed by Han et al. [Han04]

• Two phases:
  – Scan the transaction database, in order to find the frequent items of each session and store them in descending frequency order
  – Construct the FP-tree
FP-Growth example (1/2)

- **Transaction table**

<table>
<thead>
<tr>
<th>TID</th>
<th>Itemset</th>
<th>Ordered frequent items (min freq=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>f, a, c, a, d, g, i, a, m, c, p</td>
<td>f, c, a, m, p</td>
</tr>
<tr>
<td>200</td>
<td>a, b, c, f, c, l, a, m, o</td>
<td>f, c, a, b, m</td>
</tr>
<tr>
<td>300</td>
<td>b, f, h, j, o, f</td>
<td>f, b</td>
</tr>
<tr>
<td>400</td>
<td>b, c, k, s, p, c, b</td>
<td>c, b, p</td>
</tr>
<tr>
<td>500</td>
<td>a, c, f, c, e, l, f, p, m, n, a</td>
<td>f, c, a, m, p</td>
</tr>
</tbody>
</table>
FP-Growth example (2/2)

• The FP-tree construction process
GP-Close (1/2)

• A frequent pattern mining algorithm, designed by Jiang et. al. [Jia06, Jia07] to work on transaction data
  – Makes use of a taxonomy
• Deals with the over-generalization problem:
  – If any items are subsumed by other, more specialized ones, remove the former from the closure enumeration tree
• The aforementioned problem is proven to be equivalent to closed pattern discovery
GP-Close (2/2)

- Find all frequent 1-item sets
  - Can be either taxonomy categories or items
- Sort them in a length decreasing – support increasing manner
- Generate the closed closures of length $k$, based on the closures of length $k-1$ and the children of the root

- Two pruning techniques (Subtree pruning and Child-Closure pruning) manage to reduce the pattern search space
Outline

• Motivation
• Preliminaries
• The FGP Algorithm
  – Algorithm Outline
  – Algorithm Example
• Experimental Evaluation
• Future Work
• References
The FGP algorithm (1/2)

• Combines the forces of FP-Growth and GP-Close
  – GP-Close input is the FP-tree and the taxonomy (and NOT the transaction database)

• Requires modifications of the aforementioned algorithms
An outline of FGP

Transaction Database (Log Files) → Modified FP-Growth → WFP-Tree

Taxonomy → Modified GP-Close → Recommendations
The FGP algorithm (2/2)

• Scan the transaction database and construct the so-called WFP-tree (including weight information in the nodes)
• Create generalized 1-item sets using the taxonomy
  – Sort 1-item sets in increasing support order
  – Implement a modified Child-Closure pruning
• Combine 1-item sets to create the complete closure enumeration tree
  – Implement a modified Subtree pruning
• Produce the closure enumeration tree levels, until it is not possible to expand the tree any more
FGP Example (1/6)

- Input taxonomy

```
0
   / \       Taxonomy
  1   2
 / \   / \  
11 13 17 22 23 24
j a b i s d e k m p l f g h c o n
```

```
## FGP Example (2/6)

- **Transaction Database (including sessionized Web log data)**

<table>
<thead>
<tr>
<th>TID</th>
<th>Session items (PID, hits)</th>
<th>Total hits/session</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>(a,3), (c,2), (f,1), (d,1), (g,1), (i,1), (m,1), (p,1)</td>
<td>11</td>
</tr>
<tr>
<td>200</td>
<td>(a,2), (c,2), (b,1), (f,1), (l,1), (m,1), (o,1)</td>
<td>9</td>
</tr>
<tr>
<td>300</td>
<td>(f,2), (b,1), (h,1), (j,1), (o,1)</td>
<td>6</td>
</tr>
<tr>
<td>400</td>
<td>(b,2), (c,2), (k,1), (s,1), (p,1)</td>
<td>7</td>
</tr>
<tr>
<td>500</td>
<td>(a,2), (f,2), (c,2), (e,1), (l,1), (p,1), (m,1), (n,1)</td>
<td>11</td>
</tr>
</tbody>
</table>
FGP Example (3/6)

- The WFP-tree
FGP Example (4/6)

- Create generalized 1-item sets using the taxonomy
  - Find all leaf nodes, which are descendants of this node
  - Scan the WFP-tree
    - If there is a path containing all previously computed items, add the support of the corresponding items to the total value
  - Subtract multiple occurrences of the same node in a path in the support counting
FGP Example (5/6)

- Find frequent k-item sets ($k>1$)
  - Find all descendants of the items included in this closure enumeration tree node, that are taxonomy leaves
  - Find the latter in the WFP-tree
  - Check existence of a path containing a representative from each category
    - If there is such a path, increment support of each element
  - Total weight = sum of weights of all nodes in this path
FGP Example (6/6)

- Expand 1-item sets
- Prune nodes that have the same support as one of their direct ancestors
Outline

• Motivation
• Preliminaries
• The FGP Algorithm
• Experimental Evaluation
• Future Work
• References
Evaluation of FGP (1/5)

- Data set statistics

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of files</td>
<td>31</td>
</tr>
<tr>
<td>Average number of page hits per day</td>
<td>8708</td>
</tr>
<tr>
<td>Average number of sessions per day</td>
<td>882</td>
</tr>
<tr>
<td>Average session length (in page hits)</td>
<td>8.5</td>
</tr>
<tr>
<td>Average time for CE tree creation</td>
<td>17.2 sec</td>
</tr>
<tr>
<td>Average number of k-item sets per day (FP-Growth)</td>
<td>7</td>
</tr>
<tr>
<td>Average number of generalized k-item sets per day</td>
<td>56</td>
</tr>
<tr>
<td>Overall maximum rule length</td>
<td>8</td>
</tr>
</tbody>
</table>
Evaluation of FGP (2/5)

• Observation: The utilization of the taxonomic information results in the increase in the number of generated rules (56 vs. 7)

• Typically, we should measure the number of recommendations selected by the users

• We do not have real-time feedback on the recommendations
Evaluation of FGP (3/5)

• Use FGP to generate recommendations for a day and then validate them using transaction information of the next day:
  – FGP produces generalized $k$-item sets.
  – If a user has viewed articles belonging to the $k-1$ categories of the set, recommend the $k^{th}$ item (or items from the $k^{th}$ category)

• In our experiments, we find generalized frequent item sets of day $k$ ($0<=k<=30$) and attempt to locate them in the logs of day $k+1$
Evaluation of FGP (4/5)

- Two metrics of validity
  - session coverage (SC) = validSessions/allSessions

- Valid sessions are those that match at least one rule
Evaluation of FGP (5/5)

– Count the number of rules that a session validates

– In both metrics FGP is better than FP-Growth
Outline

- Motivation
- Preliminaries
- The FGP Algorithm
- Experimental Evaluation
- Future Work
- References
Future Work

- We focused on combining an ARM and a generalized pattern mining algorithm
- The order, in which pages are visited in a session are of no importance
  - Modify FGP to perform frequent generalized sequence mining
- Extend the algorithm to support multiple category assignments for a page (useful in blog aggregators)
  - This implies the creation of a taxonomy graph instead of a tree
- In the case of social data, user tags should be mapped to predefined taxonomies in order to apply FGP
- Implement a real-time application and apply our recommendation engine. This will allow us to perform a user-based evaluation
Outline

• Motivation
• Preliminaries
• The FGP Algorithm
• Experimental Evaluation
• Future Work
• References
References

- **FP-Growth**
  - [Han04] J. Han, J. Pei, Y. Yin, R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, 8, p. 53-87, Kluwer Academic Publishers, 2004

- **GP-Close**
  - [Jia07] T. Jiang, A. Tan, K. Wang, Mining Generalized Associations of Semantic Relations from Textual Web Content, Transactions on Knowledge and Data Engineering, Vol.19, No.2, February 2007
Any questions?

P. Giannikopoulos, cst04006@uop.gr
I. Varlamis, varlamis@aueb.gr
M. Eirinaki, magdalini.eirinaki@sjsu.edu

Thank you!