Mining Top-K Association Rules

Philippe Fournier-Viger[1]

Cheng-Wei Wu[2]

Vincent Shin-Mu Tseng[2]

[1]University of Moncton, Canada

[2]National Cheng Kung University, Taiwan

AI 2012 – 28 May 2012

Introduction

- A fundamental data mining task with wide applications is **association rule mining** [Agrawal93].
- It consists of finding interesting associations between items in a **transaction database**.
- A **transaction database** is a set of **transactions**, where each transaction is a set of items (an itemset).
- For example:

ID	Transactions
t_1	{a, b, c, e, f, g}
t_2	{a, b, c, d, e, f}
t_3	{a, b, e, f}
t_4	{b, f, g}

Association Rules

- **Association rule:** an association between two itemsets X→Y.
  - **Support :** the percentage of transactions of a database where the rule occurs.
  - **Confidence :** the support of the rule divided by the support of its antecedent.
- **For example**:

ID	Transactions
t_1	{a, b, c, e, f, g}
t_2	{a, b, c, d, e, f}
t_3	{a, b, e, f}
t_4	{b, f, g}

{a, b} → {c} support = 0.5 confidence = 0.66

3

Association Rule Mining

- **Goal**?
  - Discovering all rules that have a support and confidence respectively higher or equal to user-defined thresholds *minsup* and *minconf*.
- **How?**
  1. Mining frequent itemsets with an algorithm such as Apriori, FPGrowth and Hmine.
  2. For each frequent itemset X, select an itemset $P \subseteq X$ to generate a rule of the form P →X – P. Calculate the support and confidence.

4

Example

***minSup*= 0.5** and ***minConf* = 0.5**

ID	Transactions
t_1	{a, b, c, e, f, g}
t_2	{a, b, c, d, e, f}
t_3	{a, b, e, f}
t_4	{b, f, g}

A transaction database

ID	Rules	Support	Confidence
r_1	{a}→ {b}	0.75	1
r_2	{a}→ {c, e, f}	0.5	0.6
r_3	{a, b}→ {e, f}	0.75	1
…	…	…	…

Some association rules found

5

How to choose the thresholds?

- If set too high,
  - generate too few results, omitting valuable information.
- If set too low,
  - algorithms can generate an extremely large amount of results,
  - algorithms can become very slow.
- A major problem because
  - users have limited resources for analyzing the results
  - fine tuning the parameters is time-consuming.

6

The problem of setting *minsup*

- **Scenario:** A user wants to discover the top 1000 rules from a database and do not want to find more than 2000 rules.
- For **Connect**, the range of minsup values that will satisfy the user is 0.5060 to 0.5052.
- A user having no a priori knowledge of the database has only a 0.08 % chance of selecting a **minsup** value that will make him satisfied.
- Too high, not enough rules.
- Too low, performance deteriorates.

7

Our proposal

- Redefining the **problem of association rule mining** as **mining the top-*k* association rules**.
- **Two parameters:**
  - *k* is the number of rules to be generated.
  - *minconf*
- **Related works**: some works have used the term "top-*k* association rules". But they are applied to mining streams or mining non-standard rules. [Webb05, You2010]

8

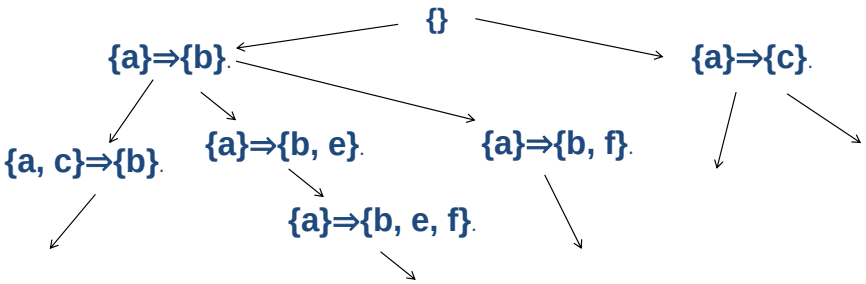
Challenges

- An algorithm for top-k association mining cannot use *minsup* to prune the search space.
- **Large search space**: if *d* different items, up to $3^d - 2^d + 1$ rules.
- Adapting the two-steps process for association mining would be inefficient because we would need to mine itemsets with *minsup* = 0 in the first step.
- So how to define an algorithm?

9

TopKRules

Find larger rules by recursively scanning the database for adding a single item at a time to the left or right part of each rule (these processes are called **left** and **right expansions**).



TopKRules (2)

Main idea

- set *minsup* = 0.
- use left/right expansions for rule generation.
- keep a set *L* that contains the current top-*k* rules found until now.
- when *k* rules are found, raise *minsup* to the lowest support of the rules in *L*.
- after that, for each rules added to *L*, raise the *minsup* threshold.

11

TopKRules (3)

- The resulting algorithm has poor execution time because the search space is too large.
- **Observation:** if we can find rules with higher support first, we can raise *minsup* more quickly and prune the search space.
- **How to define what is the most promising?** Our experiment show that the support is a good choice.
- We added a set R containing the *k* rules having the highest support.

12

TopKRules (4) - optimizations

- We found that the choice of data structures for implementing L and R is also very important:
  - *L* : fibonnaci heap : O(1) amortized time for insertion and minimum, and O(log(n)) for deletion.
  - *R*: red-black tree: O(log(n)) worst case time complexity for insertion, deletion, min and max.
- Merging database scans

13

Experimental evaluation

- 4 real-life datasets and 1 synthetic dataset

Datasets	Number of transactions	Number of distinct items	Average transaction size
Chess	3,196	75	37
Connect	67,557	129	43
T25I10D10K	10,000	1,000	25
Mushrooms	8,416	128	23
Pumsb	49,046	7,116	74

14

Results – influence of *k*

- Execution time an maximum memory usage grow linearly with *k*.

Table 2: Results for *minconf* = 0.8 and *k* = 100, 1000 and 2000

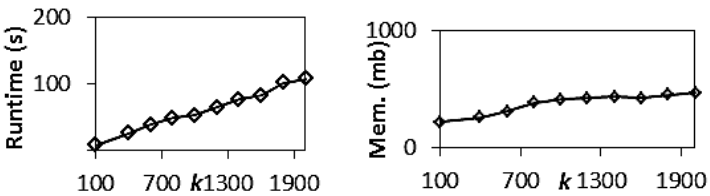
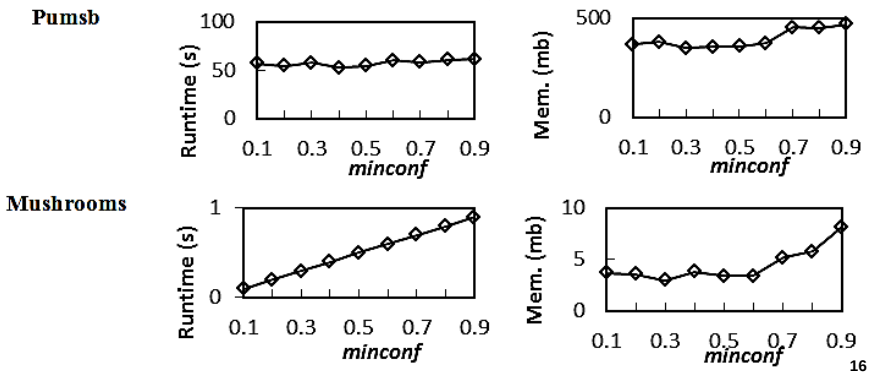


Fig. 6. Detailed results of varying *k* for the *Pumsb* dataset

15

Results – influence of *minconf*

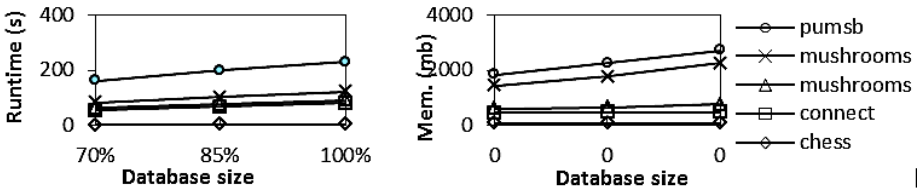
- Execution time and memory usage increase when *minconf* increases because more rules have to be generated.
- How much it increases, depends on the dataset.



16

Results – influence of database size

- Execution time and memory increases slowly if the number of rules stay more or less the same.



Performance comparison

- We compared with an implementation of association rule generation with FPGrowth, for optimal *minsup* selection.

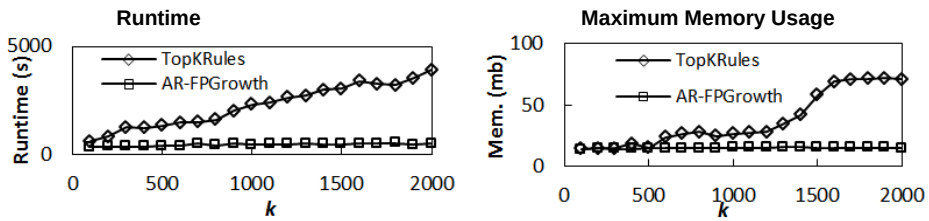


Fig. 9. Performance comparison for optimal *minsup* values for *chess*.

Performance comparison (2)

Table 4: Interval of *minsup* values to find the top 1000 to 2000 rules for each dataset

Datasets	*minsup* for k=1000	*minsup* for k=2000	Interval size
Chess	0.9415	0.9324	0.0091
Connect	0.5060	0.5052	0.0008
T25I10D10K	0.0120	0.0100	0.0020
Mushrooms	0.4610	0.4454	0.0156
Pumsb	0.6639	0.6594	0.0044

- When *minsup* is chosen optimally, FPGrowth has better performance.
- However, setting *minsup* is very difficult.
- If *minsup* is set too low, FPGrowth will not find any rule.
- If *minsup* is set too high, too many rules will be found and the performance deteriorates.

19

Conclusion

- We proposed an algorithm that let the user set *k*, the number of rules to be found.
- Excellent scalability: execution time linearly increases with *k*.
- The algorithm has no problem running in reasonable time and memory limits for *k* values of up to 5000 for all datasets.

20

Thank you. Questions?

SPMF

Open source Java data mining software, 46 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Thanks to the FQRNT funding programs.

Fonds de recherche sur la nature et les technologies
Québec