The CMRules Algorithm

Philippe Fournier-Viger
http://www.philippe-Fournier-viger.com

Fournier-Viger, P., Faghihi, U., Nkambou, R., Mephu Nguifo, E. (2012). **CMRules: Mining Sequential Rules Common to Several Sequences**. Knowledge-based Systems, Elsevier, 25(1): 63-76.

Source code and datasets available in the SPMF library

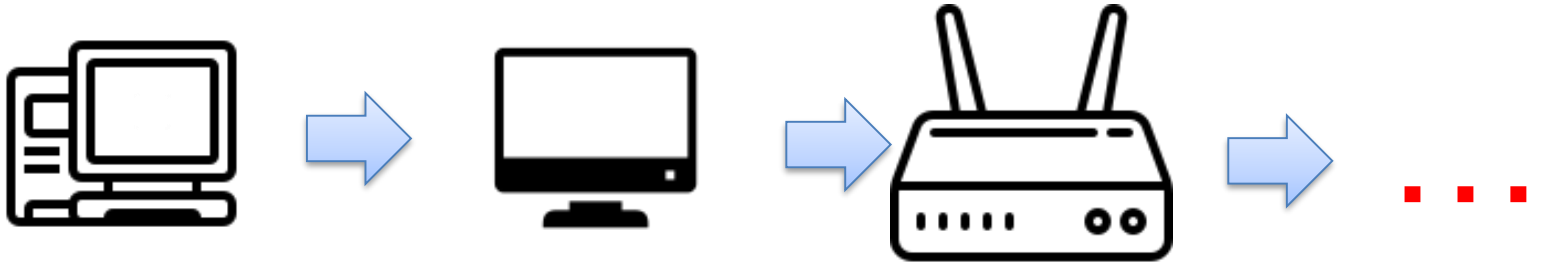
Introduction

- More and more data!
- A need to analyze data to find **interesting patterns**
- **Pattern mining**: using algorithms to find interesting patterns in data.
- An important type of data is **sequences**.
- Today, we will discuss how to analyze sequences to find **sequential rules using the CMRules algorithm**.

What is a **discrete sequence**?

Sequence: an ordered list of symbols

Sequence of purchases



Sequence of words

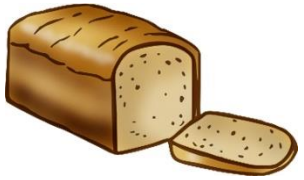
Definition: Items

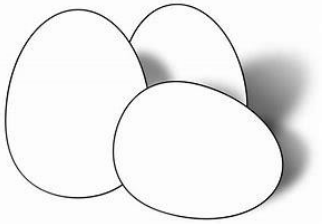
Let there be a **set** of **items** (symbols) called I.

Example: $I = \{a, b, c, d, e, f, g\}$

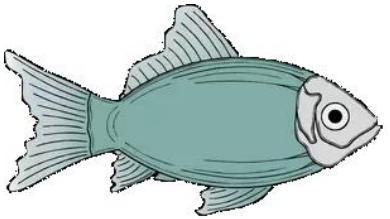
a = apple 

d = dattes 

b = bread 

e = eggs 

c = cake 

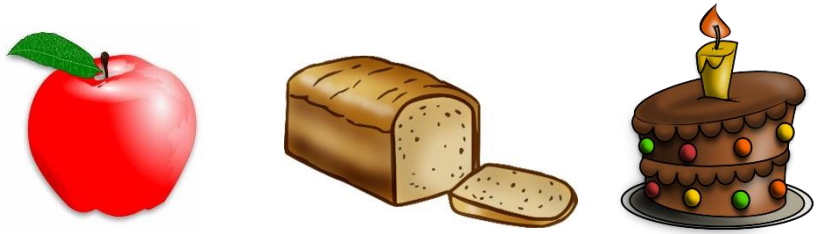
f = fish 

g = grapes 

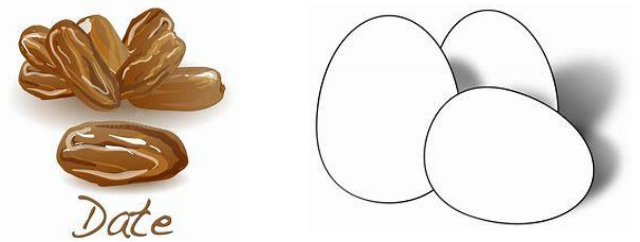
Definition: Itemset

An itemset is a set of **items** that is a subset of I.

Example: $\{a, b, c\}$ is an itemset containing 3 items



$\{d, e\}$ is an itemset containing 2 items

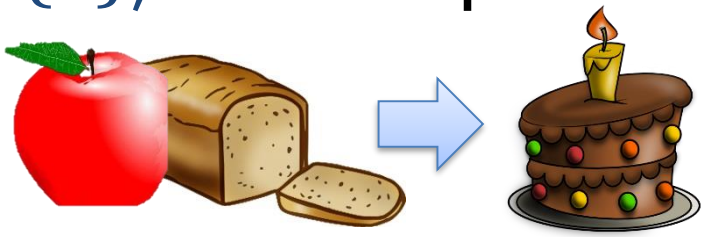


- Note: an itemset cannot contain a same item twice.
- An itemset having k items is called a *k-itemset*.

Definition: Sequence

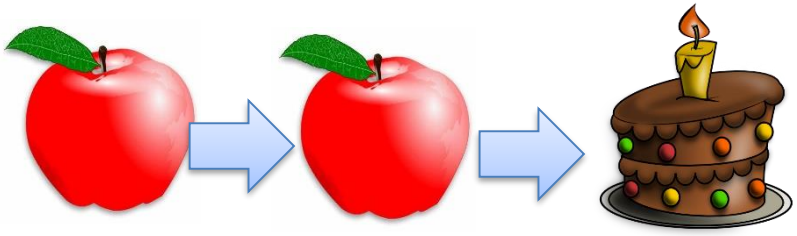
A **discrete sequence** S is a an ordered list of itemsets $S = \langle X_1, X_2, \ldots, X_n \rangle$ where $X_j \subseteq I$ for any $j \in \{1,2..n\}$

Example 1: $\langle \{a, b\}, \{c\} \rangle$ is a sequence containing two itemsets.



It means that a customer purchased $apple$ and $bread$ at the same time and then purchased $cake$.

Example 2: $\langle \{a\}, \{a\}, \{c\} \rangle$



Definition: Sequence Database

- A **sequence database** is one or more sequences.

SID	sequence
1	<{a}, {a,b,c} {a, c} {d} {c, f}>
2	<{a, d}, {c} {b, c} {a, e}>
3	<{e, f}, {a, b} {d, f} {c}, {b}>
4	<{e}, {g}, {a, f} {c} {b}, {c}>

- Here we have four sequences.
- Each sequence has a unique sequence identifier (SID)

We want to find patterns revealing strong relationships between items in sequences! →

Sequential Rule Mining

The goal is to discover rules in sequences.

Partially-Ordered Sequential rule: Rule of the form $X \rightarrow Y$, where X and Y are itemsets that are **unordered**, non empty and disjoint.

Example: {a,b} $\rightarrow$ {f,g} is a sequential rule

Sequential Rule Mining

The goal is to discover rules in sequences.

Partially-Ordered Sequential rule: Rule of the form $X \rightarrow Y$, where X and Y are itemsets that are **unordered**, non empty and disjoint.

Example: {a,b} $\rightarrow$ {f,g} is a sequential rule

Interpretation:
If we observe **a** and **b** (in any order), they will be followed by **f** and **g** (in any order)

Sequential Rule Mining

Partially-Ordered Sequential rule: Rule of the form $X \rightarrow Y$, where X and Y are itemsets that are **unordered**, non empty and disjoint.

More examples:

{a,b} $\rightarrow$ {a} is **NOT** a sequential rule

{} $\rightarrow$ {f} is **NOT** a sequential rule

How to find interesting sequential rules?

- A sequential rule X→Y has **two properties**:
  - **Sequential support sup(X→Y):** the number of sequences where X occurs before Y, divided by the number of sequences.
  - **Sequential confidence conf(X→Y):** the number of sequences where X occurs before Y, divided by the number of sequences where X occurs.
- **The task**: finding all **valid rules**, rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf* (Fournier-Viger, 2010).

An example of Sequential Rule Mining

Let say that *minSup*= 0.5 and *minConf*= 0.5:

ID	Sequences
seq1	$\{a, b\},\{c\},\{f\},\{g\},\{e\}$
seq2	$\{a, d\},\{c\},\{b\},\{a, b, e, f\}$
seq3	$\{a\},\{b\},\{f\},\{e\}$
seq4	$\{b\},\{f, g\}$

A sequence database

→

ID	Rule	Support	Confidence
r1	$\{a, b, c\} \Rightarrow \{e\}$	0.5	1.0
r2	$\{a\} \rightarrow \{c, e, f\}$	0.5	0.66
r3	$\{a, b\} \rightarrow \{e, f\}$	0.75	1.0
r4	$\{b\} \rightarrow \{e, f\}$	0.75	0.75
r5	$\{a\} \rightarrow \{e, f\}$	0.75	1.0
r6	$\{c\} \rightarrow \{f\}$	0.5	1.0
r7	$\{a\} \rightarrow \{b\}$	0.5	0.66
…	…	…	…

Some rules found

Not an easy problem!

- Let's say that there are *r* distinct items in a sequence database, that is $|I| = r$.
- We can prove that the number of potential sequential rules is:

$$\sum_{k=1}^{r-1}\left[\binom{r}{k}\times\sum_{j=1}^{r-k}\binom{r-k}{j}\right]=3^r-2^{r+1}+1$$

- Thus, we do not want to explore all the possibilities!

The CMRules algorithm

- This is the first algorithm for this problem.
- The key observation is that sequential rules are a type of association rules.
- CMRules perform two phases:
  - CMRules first applies an association rule mining algorithm such as Apriori to find association rules.
  - Then, CMRules eliminates association rules that are not sequential rules.

Association Rule Mining

- **A transaction database D** is a set of transactions $T=\{t_1,t_2...t_m\}$ where $t_1,t_2,...,t_n \subseteq I$.
- **Association rule:** $X \rightarrow Y$ such that $X,Y \subseteq I$, $X \cap Y=\emptyset$,
- The **support** of a rule $X \rightarrow Y$ is defined as $sup(X \cup Y) / |D|$.
  The **confidence** of a rule is defined as $conf(X \rightarrow Y) = sup(X \cup Y) / sup(X)$.
- **Association rule mining**: finding all rules such that their support and confidence are no less than some thresholds *minsup* and *minconf*.

ID	transactions
1	a b c e f g
2	a b c d e f
3	a b e f
4	b f g

(2) Association rule mining

Association rules

ID	rule	Sup.	Conf.
1	c → a b e f	0.5	1.0
2	a b c → e	0.5	1.0
3	a → c e f	0.5	0.6
4	a b → e f	0.75	1.0
5	b → e f	0.75	0.75
6	a → e f	0.75	1.0
7	c → f	0.5	1.0
8	a → b	0.75	1.0
	...	...	...

The Relationship between Association Rules and Sequential Rules

- A **sequence database S can be transformed into a transaction database S'** by removing time information.
- Each sequential rule r: $X \Rightarrow Y$ of S has a corresponding association rule r': $X \rightarrow Y$ in S'.
- The relationships holds:
  - $sup(r') \geq sup(r)$
  - $conf(r') \geq conf(r)$,

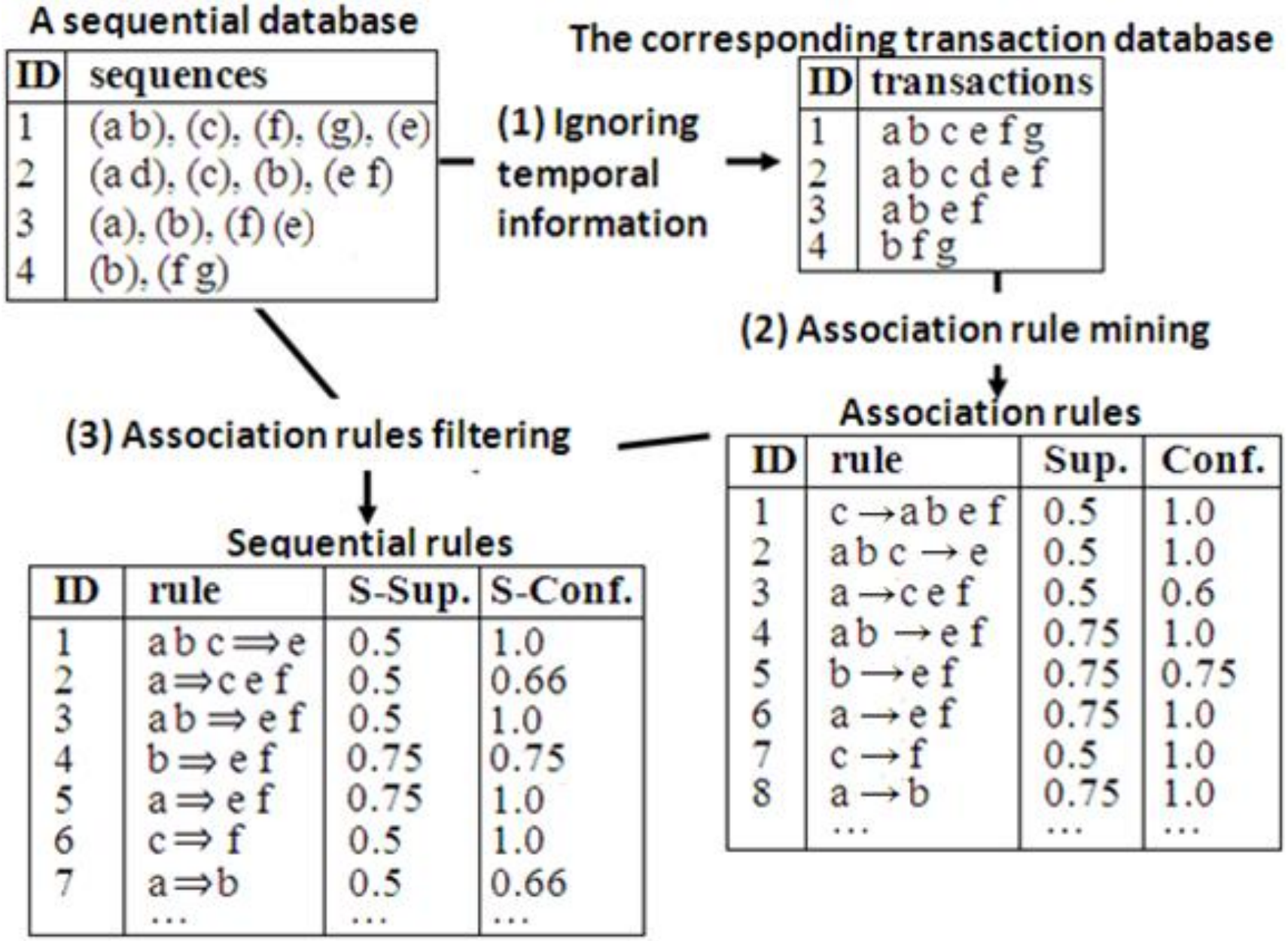
The CMRules algorithm

INPUT : a sequence database, *minSup*, *minConf*
OUTPUT : the set of all sequential rules
PROCEDURE:

1. Consider the sequence database as a transaction database
2. Find all association rules from the transaction database by applying an association rule mining algorithm such as Apriori (Agrawal et al., 1993).
3. Scan the original sequence database to calculate the sequential support and sequential confidence of each association rule found in the previous step. Eliminate each rule *r* such that:
   a. Sup(r) < *minSup*
   b. Conf(r) < *minConf*
4. Return the set of rules

see the article for the proof of correctedness.

A Sample Execution of the CMRules algorithm



Optimizations

- **How to calcultate** the support of a sequential rule X→Y ?
  - The naïve approach is to check all sequences.
  - Association rule mining algorithms first find frequent itemsets X and Y and then generate rules of the form: X→ Y-X
  - Algorithms such as AprioriTID, Eclat, H-Mine, etc. can annotate itemset X and Y with the sequences that contains them.
  - If we use such algorithm, we can only check sequence containing X for calculating sup(X → Y) .
  - This can improve performance by up to 50 %.
- Also, we don't need to keep the association rules. We can discard each rule immediately after it is found if it is not a sequential rule. This reduces memory consumption.

Analysis of the Time Complexity

- Step 1: converting a sequence database in a transaction database is done in linear time.
- Step 2: association rule mining. Two substeps:
  1. Discovering frequent itemsets
     $O(2^r)$ r= number of diff. items
  2. Generating association rules from frequent itemsets: less costly, thus can be ignored.
- Step 3: calculating sequential conf. and support.
  For each rule, **best case**: $|S| \times minsup$ sequences to check, **worst case**: $|S|$ sequences to check. Checking if a rule is included in a sequence is done in linear time.

Conclusion

Summary

- **CMRules**: an algorithm for mining sequential rules common to several sequences.
- CMRules can discover sequential rules and association rules at the same time
- Advantage: can reuse the code of existing association rule mining algorithms
- Some more efficient algorithms have been proposed such as RuleGrowth, ERMiner…

Source code and dataset in the **SPMF library**