Frequent Itemset Mining (2)

Course Notes 4

Nadjib Lazaar
University of Paris-Saclay
lazaar@lisn.fr

---

1 LCM Algorithm: Linear-Time Closed Itemset Mining

The **LCM Algorithm** (Linear-Time Closed Itemset Mining) is an efficient algorithm designed for discovering frequent closed itemsets in large transactional datasets [Uno et al., 2004]. A closed itemset is an itemset for which there is no superset with the same support count (see Definition **??**). The LCM algorithm aims to efficiently generate closed itemsets, which can provide more compact and meaningful insights into the data compared to regular frequent itemsets.

1.1 LCM Algorithm Mechanics

LCM algorithm is characterized by its efficient traversal and data representation strategies. Below, we describe its key operational principles:

1. **Lexicographic Order Traversal:** The candidate itemsets are checked in lexicographic order by performing a depth-first traversal of the prefix tree. This systematic exploration ensures that all frequent closed itemsets are discovered without redundancy.
2. **Recursive Processing of Conditional Transaction Databases:** The algorithm processes a reduced transaction database at each recursive step. By step-by-step elimination of items from the original dataset, the conditional transaction databases are recursively created and mined for further closed itemsets.
3. **Dual Data Representation:** The algorithm maintains both a horizontal and a vertical representation of the transaction database in parallel:
   - The **horizontal representation** stores transactions as lists of items.
   - The **vertical representation** stores each item alongside the list of transaction IDs in which it appears.

4. **Filtering with Vertical Representation:** During the recursion, the vertical representation is used to filter the transactions based on the chosen split item. This filtering ensures that only relevant transactions are considered in subsequent steps, significantly reducing the search space.

5. **Updating for Recursion with Horizontal Representation:** The horizontal representation is employed to generate the vertical representation for the next recursive step. Unlike the Eclat algorithm, which uses intersection operations for this purpose, LCM avoids intersection computations, making it more efficient.

1.2 Notations and Definitions

The *prefix tree* is a hierarchical data structure used to represent relationships among subsets of items (itemsets) efficiently. Each node in the tree corresponds to an item, and each path from the root to a node represents a unique itemset. Itemsets sharing a common prefix also share the same initial path in the tree, enabling efficient organization of data and systematic exploration, especially in frequent pattern mining algorithms.

Definition 1 (Prefix in the Prefix Tree) *Let $P = \{a_1, \ldots, a_m\}$ be an itemset of m items, where $a_i \in \mathcal{I}$ and $a_i < a_{i+1}$ according to a fixed order on the items (e.g., lexicographic order).*

We define:

- ***Tail:*** *The* tail *of P, denoted by $\texttt{tail}(P)$, is the last item in P, i.e., $\texttt{tail}(P) = a_m$.*
- ***Sub-itemsets:*** *The sub-itemset of P up to the jth item, denoted by $P(a_j)$, is defined as:*
$$P(a_j) = \{a_1, \ldots, a_j\}.$$
- ***Prefix:*** *The* prefix *of P in the prefix tree, denoted by* $\text{prefix}(P)$*, is defined as:*
$$\text{prefix}(P) = P \setminus \texttt{tail}(P) = P(\texttt{tail}(P)) = \{a_1, \ldots, a_{m-1}\}.$$

Definition 2 (Closure) *A set S has a closure under an operation f if:*

- $\forall x \in S,\ f(x) \in S.$

In this case, we say that the set S is closed under f.

A closure operation is:

- ***Increasing (or extensive):*** *The closure of an object contains the object itself, i.e., $S \subseteq closure(S)$.*
- ***Idempotent:*** *The closure of the closure of an object is the closure itself, i.e., $closure(closure(S)) = closure(S)$.*
- ***Monotone:*** *If $X \subseteq Y$, then $closure(X) \subseteq closure(Y)$.*

Definition 3 (Itemset Closure) *The closure of an itemset P, denoted as* $\texttt{closure}(P)$*, is the intersection of all transactions t in the cover of P. Formally, we define the closure of P as:*

$$\texttt{closure}(P) = \bigcap_{t \in \texttt{cover}(P)} t$$

where $\texttt{cover}(P)$ *represents the set of transactions that contain the itemset P, and t is a transaction in this cover.*

An itemset P is said to be closed *if it is equal to its closure, i.e.,*

$$P = \texttt{closure}(P).$$

1.3 LCM Mechanism

The mechanism of the LCM algorithm revolves around the concept of **closure extension**, which allows for the efficient construction of closed itemsets.

1.3.1 Closure Extension: The Rule

The **closure extension** rule can be summarized as follows:

- **Start with a closed itemset** P. We can extend it by adding a new item p_i to P, forming a new candidate itemset $P' = P \cup \{p_i\}$.
- After adding the new item, we compute the **closure** of the extended itemset P', $\texttt{closure}(P')$, which we recall that is the set of items common to all transactions in $\texttt{cover}(P')$, i.e. the intersection of all transactions covering P'.
- If the closure of P' equals P' itself, meaning P' is **closed**, then P' becomes a valid closed itemset.

1.3.2 Step-by-Step Process in LCM

The process in the LCM algorithm can be described as follows:

1. Start with a frequent closed itemset P.
2. Extend P by adding a new item p_i from the set of items in the dataset, forming a candidate itemset $P' = P \cup \{p_i\}$.
3. Calculate the closure of P' by finding the intersection of all transactions that contain P':
$$\text{closure}(P') = \bigcap_{t \in \texttt{cover}(P')} t$$
where $\texttt{cover}(P')$ is the set of transactions that contain P'.
4. Recursively repeat the process for each new valid closed itemset until no more can be found.

Algorithm 1: LCM

InOut: P: Closed Frequent Itemset;
In: α: minsup

`print`(P)
foreach $p_i > \texttt{tail}(P)$ **do**
  if $freq(P \cup \{p_i\}) \geq \alpha$ **then** LCM(`closure`($P \cup \{p_i\}$, α)

1.3.3 Why Closure Extension Works

- **Monotonicity:** If P is closed, then adding items to P and computing the closure of the result will never yield an itemset that is not closed, as long as the closure operation is maintained.
- **Efficient Construction:** The closure extension rule allows the LCM algorithm to systematically discover all closed itemsets by incrementally extending smaller closed itemsets and checking for closure, without having to check every possible combination of items directly.

Theorem 1 *[Uno et al., 2004] Let α be a minimum support. Algorithm LCM enumerates, given the root closed itemset $\emptyset$ =* `closure`($\emptyset$)*, all frequent closed itemsets in linear time in the number of frequent closed itemsets in .*

References

[Uno et al., 2004] Uno, T., Kiyomi, M., and Arimura, H. (2004). LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004.*