CHAPTER 9 Summarizing Itemsets

The search space for frequent itemsets is usually very large and it grows exponentially with the number of items. In particular, a low minimum support value may result in an intractable number of frequent itemsets. An alternative approach, studied in this chapter, is to determine condensed representations of the frequent itemsets that summarize their essential characteristics. The use of condensed representations can not only reduce the computational and storage demands, but it can also make it easier to analyze the mined patterns. In this chapter we discuss three of these representations: closed, maximal, and nonderivable itemsets.

9.1 MAXIMAL AND CLOSED FREQUENT ITEMSETS

Given a binary database $\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$, over the tids $\mathcal{T}$ and items $\mathcal{I}$, let $\mathcal{F}$ denote the set of all frequent itemsets, that is,

$$\mathcal{F} = \{X \mid X \subseteq \mathcal{I} \text{ and } sup(X) \geq minsup\}$$

Maximal Frequent Itemsets

A frequent itemset $X \in \mathcal{F}$ is called *maximal* if it has no frequent supersets. Let $\mathcal{M}$ be the set of all maximal frequent itemsets, given as

$$\mathcal{M} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X, \text{ such that } Y \in \mathcal{F}\}$$

The set $\mathcal{M}$ is a condensed representation of the set of all frequent itemset $\mathcal{F}$, because we can determine whether any itemset X is frequent or not using $\mathcal{M}$. If there exists a maximal itemset Z such that $X \subseteq Z$, then X must be frequent; otherwise X cannot be frequent. On the other hand, we cannot determine $sup(X)$ using $\mathcal{M}$ alone, although we can lower-bound it, that is, $sup(X) \geq sup(Z)$ if $X \subseteq Z \in \mathcal{M}$.

Example 9.1. Consider the dataset given in Figure 9.1a. Using any of the algorithms discussed in Chapter 8 and $minsup = 3$, we obtain the frequent itemsets shown in Figure 9.1b. Notice that there are 19 frequent itemsets out of the $2^5 - 1 = 31$ possible nonempty itemsets. Out of these, there are only two maximal itemsets,

Tid	Itemset
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

(a) Transaction database

sup	Itemsets
6	B
5	E, BE
4	$A, C, D, AB, AE, BC, BD, ABE$
3	$AD, CE, DE, ABD, ADE, BCE, BDE, ABDE$

(b) Frequent itemsets ($minsup = 3$)

Figure 9.1. An example database.

$ABDE$ and BCE. Any other frequent itemset must be a subset of one of the maximal itemsets. For example, we can determine that ABE is frequent, since $ABE \subset ABDE$, and we can establish that $sup(ABE) \geq sup(ABDE) = 3$.

Closed Frequent Itemsets

Recall that the function $\mathbf{t}: 2^{\mathcal{I}} \rightarrow 2^{\mathcal{T}}$ [Eq. (8.2)] maps itemsets to tidsets, and the function $\mathbf{i}: 2^{\mathcal{T}} \rightarrow 2^{\mathcal{I}}$ [Eq. (8.1)] maps tidsets to itemsets. That is, given $T \subseteq \mathcal{T}$, and $X \subseteq \mathcal{I}$, we have

$$\mathbf{t}(X) = \{t \in \mathcal{T} \mid t \text{ contains } X\}$$

$$\mathbf{i}(T) = \{x \in \mathcal{I} \mid \forall t \in T,\ t \text{ contains } x\}$$

Define by $\mathbf{c}: 2^{\mathcal{I}} \rightarrow 2^{\mathcal{I}}$ the *closure operator*, given as

$$\mathbf{c}(X) = \mathbf{i} \circ \mathbf{t}(X) = \mathbf{i}(\mathbf{t}(X))$$

The closure operator $\mathbf{c}$ maps itemsets to itemsets, and it satisfies the following three properties:

- *Extensive*: $X \subseteq \mathbf{c}(X)$
- *Monotonic*: If $X_i \subseteq X_j$, then $\mathbf{c}(X_i) \subseteq \mathbf{c}(X_j)$
- *Idempotent*: $\mathbf{c}(\mathbf{c}(X)) = \mathbf{c}(X)$

An itemset X is called *closed* if $\mathbf{c}(X) = X$, that is, if X is a fixed point of the closure operator $\mathbf{c}$. On the other hand, if $X \neq \mathbf{c}(X)$, then X is not closed, but the set $\mathbf{c}(X)$ is called its closure. From the properties of the closure operator, both X and $\mathbf{c}(X)$ have the same tidset. It follows that a frequent set $X \in \mathcal{F}$ is closed if it has no frequent superset *with the same frequency* because by definition, it is the largest itemset common to all the tids in the tidset $\mathbf{t}(X)$. The set of all closed frequent itemsets is thus defined as

$$\mathcal{C} = \left\{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X \text{ such that } sup(X) = sup(Y)\right\} \tag{9.1}$$

Put differently, X is closed if all supersets of X have strictly less support, that is, $sup(X) > sup(Y)$, for all $Y \supset X$.

The set of all closed frequent itemsets $\mathcal{C}$ is a condensed representation, as we can determine whether an itemset X is frequent, as well as the exact support of X using $\mathcal{C}$ alone. The itemset X is frequent if there exists a closed frequent itemset $Z \in \mathcal{C}$ such that $X \subseteq Z$. Further, the support of X is given as

$$sup(X) = \max\{sup(Z) | Z \in \mathcal{C}, X \subseteq Z\}$$

The following relationship holds between the set of all, closed, and maximal frequent itemsets:

$$\mathcal{M} \subseteq \mathcal{C} \subseteq \mathcal{F}$$

Minimal Generators

A frequent itemset X is a *minimal generator* if it has no subsets with the same support:

$$\mathcal{G} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \subset X, \text{ such that } sup(X) = sup(Y)\}$$

In other words, all subsets of X have strictly higher support, that is, $sup(X) < sup(Y)$, for all $Y \subset X$. The concept of minimum generator is closely related to the notion of closed itemsets. Given an equivalence class of itemsets that have the same tidset, a closed itemset is the unique maximum element of the class, whereas the minimal generators are the minimal elements of the class.

Example 9.2. Consider the example dataset in Figure 9.1a. The frequent closed (as well as maximal) itemsets using $minsup = 3$ are shown in Figure 9.2. We can see, for instance, that the itemsets AD, DE, ABD, ADE, BDE, and $ABDE$, occur in the same three transactions, namely 135, and thus constitute an equivalence class. The largest itemset among these, namely $ABDE$, is the closed itemset. Using the closure operator yields the same result; we have $\mathbf{c}(AD) = \mathbf{i}(\mathbf{t}(AD)) = \mathbf{i}(135) = ABDE$, which indicates that the closure of AD is $ABDE$. To verify that $ABDE$ is closed note that $\mathbf{c}(ABDE) = \mathbf{i}(\mathbf{t}(ABDE)) = \mathbf{i}(135) = ABDE$. The minimal elements of the equivalence class, namely AD and DE, are the minimal generators. No subset of these itemsets shares the same tidset.

The set of all closed frequent itemsets, and the corresponding set of minimal generators, is as follows:

Tidset	$\mathcal{C}$	$\mathcal{G}$
1345	ABE	A
123456	B	B
1356	BD	D
12345	BE	E
2456	BC	C
135	$ABDE$	AD, DE
245	BCE	CE

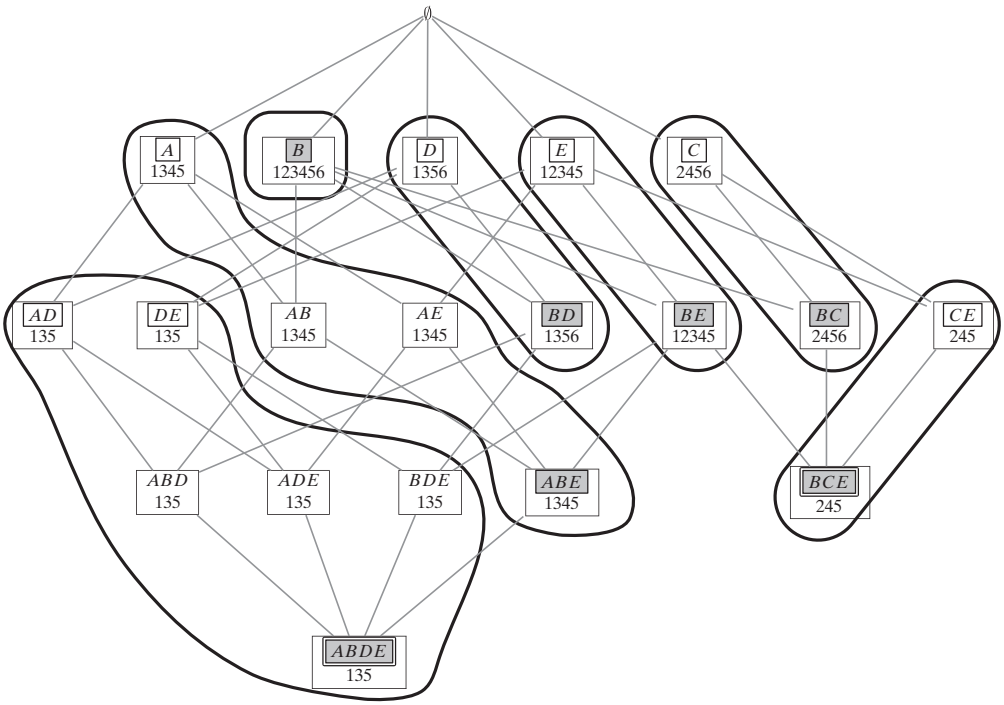


Figure 9.2. Frequent, closed, minimal generators, and maximal frequent itemsets. Itemsets that are boxed and shaded are closed, whereas those within boxes (but unshaded) are the minimal generators; maximal itemsets are shown boxed with double lines.

Out of the closed itemsets, the maximal ones are $ABDE$ and BCE. Consider itemset AB. Using $\mathcal{C}$ we can determine that

$$sup(AB) = \max\{sup(ABE), sup(ABDE)\} = \max\{4, 3\} = 4$$

9.2 MINING MAXIMAL FREQUENT ITEMSETS: GENMAX ALGORITHM

Mining maximal itemsets requires additional steps beyond simply determining the frequent itemsets. Assuming that the set of maximal frequent itemsets is initially empty, that is, $\mathcal{M} = \emptyset$, each time we generate a new frequent itemset X, we have to perform the following maximality checks

- **Subset Check:** $\nexists Y \in \mathcal{M}$, such that $X \subset Y$. If such a Y exists, then clearly X is not maximal. Otherwise, we add X to $\mathcal{M}$, as a potentially maximal itemset.
- **Superset Check:** $\nexists Y \in \mathcal{M}$, such that $Y \subset X$. If such a Y exists, then Y cannot be maximal, and we have to remove it from $\mathcal{M}$.

These two maximality checks take $O(|\mathcal{M}|)$ time, which can get expensive, especially as $\mathcal{M}$ grows; thus for efficiency reasons it is crucial to minimize the number of times these checks are performed. As such, any of the frequent itemset mining algorithms

from Chapter 8 can be extended to mine maximal frequent itemsets by adding the maximality checking steps. Here we consider the GenMax method, which is based on the tidset intersection approach of Eclat (see Section 8.2.2). We shall see that it never inserts a nonmaximal itemset into $\mathcal{M}$. It thus eliminates the superset checks and requires only subset checks to determine maximality.

Algorithm 9.1 shows the pseudo-code for GenMax. The initial call takes as input the set of frequent items along with their tidsets, $\langle i, \mathbf{t}(i)\rangle$, and the initially empty set of maximal itemsets, $\mathcal{M}$. Given a set of itemset–tidset pairs, called IT-pairs, of the form $\langle X, \mathbf{t}(X)\rangle$, the recursive GenMax method works as follows. In lines 1–3, we check if the entire current branch can be pruned by checking if the union of all the itemsets, $Y = \bigcup X_i$, is already subsumed by (or contained in) some maximal pattern $Z \in \mathcal{M}$. If so, no maximal itemset can be generated from the current branch, and it is pruned. On the other hand, if the branch is not pruned, we intersect each IT-pair $\langle X_i, \mathbf{t}(X_i)\rangle$ with all the other IT-pairs $\langle X_j, \mathbf{t}(X_j)\rangle$, with $j > i$, to generate new candidates X_{ij}, which are added to the IT-pair set P_i (lines 6–9). If P_i is not empty, a recursive call to GENMAX is made to find other potentially frequent extensions of X_i. On the other hand, if P_i is empty, it means that X_i cannot be extended, and it is potentially maximal. In this case, we add X_i to the set $\mathcal{M}$, provided that X_i is not contained in any previously added maximal set $Z \in \mathcal{M}$ (line 12). Note also that, because of this check for maximality before inserting any itemset into $\mathcal{M}$, we never have to remove any itemsets from it. In other words, all itemsets in $\mathcal{M}$ are guaranteed to be maximal. On termination of GenMax, the set $\mathcal{M}$ contains the final set of all maximal frequent itemsets. The GenMax approach also includes a number of other optimizations to reduce the maximality checks and to improve the support computations. Further, GenMax utilizes diffsets (differences of tidsets) for fast support computation, which were described in Section 8.2.2. We omit these optimizations here for clarity.

ALGORITHM 9.1. Algorithm GENMAX

```
   // Initial Call: M ← ∅, P ← {⟨i, t(i)⟩ | i ∈ I, sup(i) ≥ minsup}
   GENMAX (P, minsup, M):
1  Y ← ⋃ X_i
2  if ∃Z ∈ M, such that Y ⊆ Z then
3      return // prune entire branch
4  foreach ⟨X_i, t(X_i)⟩ ∈ P do
5      P_i ← ∅
6      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
7          X_ij ← X_i ∪ X_j
8          t(X_ij) = t(X_i) ∩ t(X_j)
9          if sup(X_ij) ≥ minsup then P_i ← P_i ∪ {⟨X_ij, t(X_ij)⟩}
10     if P_i ≠ ∅ then GENMAX (P_i, minsup, M)
11     else if ∄Z ∈ M, X_i ⊆ Z then
12         M = M ∪ X_i   // add X_i to maximal set
```

Example 9.3. Figure 9.3 shows the execution of GenMax on the example database from Figure 9.1a using $minsup = 3$. Initially the set of maximal itemsets is empty. The root of the tree represents the initial call with all IT-pairs consisting of frequent single items and their tidsets. We first intersect $\mathbf{t}(A)$ with the tidsets of the other items. The set of frequent extensions from A are

$$P_A = \{\langle AB, 1345\rangle, \langle AD, 135\rangle, \langle AE, 1345\rangle\}$$

Choosing $X_i = AB$, leads to the next set of extensions, namely

$$P_{AB} = \{\langle ABD, 135\rangle, \langle ABE, 1345\rangle\}$$

Finally, we reach the left-most leaf corresponding to $P_{ABD} = \{\langle ABDE, 135\rangle\}$. At this point, we add $ABDE$ to the set of maximal frequent itemsets because it has no other extensions, so that $\mathcal{M} = \{ABDE\}$.

The search then backtracks one level, and we try to process ABE, which is also a candidate to be maximal. However, it is contained in $ABDE$, so it is pruned. Likewise, when we try to process $P_{AD} = \{\langle ADE, 135\rangle\}$ it will get pruned because it is also subsumed by $ABDE$, and similarly for AE. At this stage, all maximal itemsets starting with A have been found, and we next proceed with the B branch. The left-most B branch, namely BCE, cannot be extended further. Because BCE is not

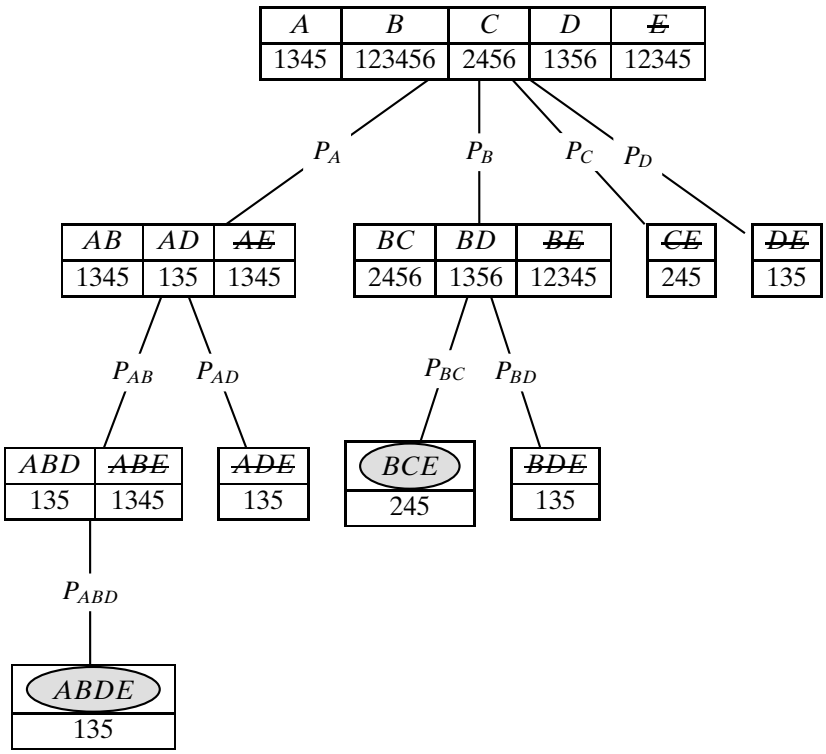


Figure 9.3. Mining maximal frequent itemsets. Maximal itemsets are shown as shaded ovals, whereas pruned branches are shown with the strike-through. Infrequent itemsets are not shown.

a subset of any maximal itemset in $\mathcal{M}$, we insert it as a maximal itemset, so that $\mathcal{M} = \{ABDE, BCE\}$. Subsequently, all remaining branches are subsumed by one of these two maximal itemsets, and are thus pruned.

9.3 MINING CLOSED FREQUENT ITEMSETS: CHARM ALGORITHM

Mining closed frequent itemsets requires that we perform closure checks, that is, whether $X = \mathbf{c}(X)$. Direct closure checking can be very expensive, as we would have to verify that X is the largest itemset common to all the tids in $\mathbf{t}(X)$, that is, $X = \bigcap_{t \in \mathbf{t}(X)} \mathbf{i}(t)$. Instead, we will describe a vertical tidset intersection based method called CHARM that performs more efficient closure checking. Given a collection of IT-pairs $\{\langle X_i, \mathbf{t}(X_i)\rangle\}$, the following three properties hold:

Property (1) If $\mathbf{t}(X_i) = \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) = \mathbf{c}(X_j) = \mathbf{c}(X_i \cup X_j)$, which implies that we can replace every occurrence of X_i with $X_i \cup X_j$ and prune the branch under X_j because its closure is identical to the closure of $X_i \cup X_j$.

Property (2) If $\mathbf{t}(X_i) \subset \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j)$ but $\mathbf{c}(X_i) = \mathbf{c}(X_i \cup X_j)$, which means that we can replace every occurrence of X_i with $X_i \cup X_j$, but we cannot prune X_j because it generates a different closure. Note that if $\mathbf{t}(X_i) \supset \mathbf{t}(X_j)$ then we simply interchange the role of X_i and X_j.

Property (3) If $\mathbf{t}(X_i) \neq \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j) \neq \mathbf{c}(X_i \cup X_j)$. In this case we cannot remove either X_i or X_j, as each of them generates a different closure.

Algorithm 9.2 presents the pseudo-code for Charm, which is also based on the Eclat algorithm described in Section 8.2.2. It takes as input the set of all frequent single items along with their tidsets. Also, initially the set of all closed itemsets, $\mathcal{C}$, is empty. Given any IT-pair set $P = \{\langle X_i, \mathbf{t}(X_i)\rangle\}$, the method first sorts them in increasing order of support. For each itemset X_i we try to extend it with all other items X_j in the sorted order, and we apply the above three properties to prune branches where possible. First we make sure that $X_{ij} = X_i \cup X_j$ is frequent, by checking the cardinality of $\mathbf{t}(X_{ij})$. If yes, then we check properties 1 and 2 (lines 8 and 12). Note that whenever we replace X_i with $X_{ij} = X_i \cup X_j$, we make sure to do so in the current set P, as well as the new set P_i. Only when property 3 holds do we add the new extension X_{ij} to the set P_i (line 14). If the set P_i is not empty, then we make a recursive call to Charm. Finally, if X_i is not a subset of any closed set Z with the same support, we can safely add it to the set of closed itemsets, $\mathcal{C}$ (line 18). For fast support computation, Charm uses the diffset optimization described in Section 8.2.2; we omit it here for clarity.

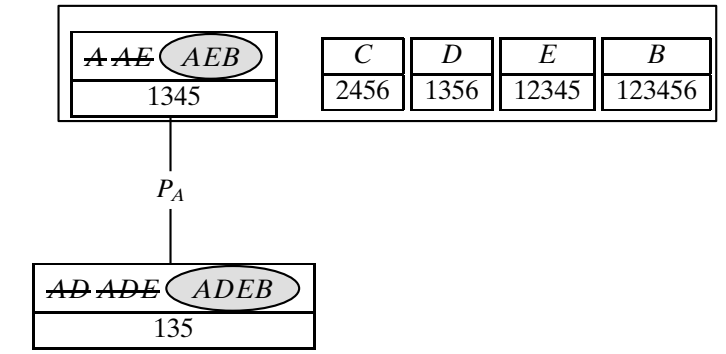
Example 9.4. We illustrate the Charm algorithm for mining frequent closed itemsets from the example database in Figure 9.1a, using *minsup* = 3. Figure 9.4 shows the sequence of steps. The initial set of IT-pairs, after support based sorting, is shown at the root of the search tree. The sorted order is A, C, D, E, and B. We first process extensions from A, as shown in Figure 9.4a. Because AC is not frequent,

ALGORITHM 9.2. Algorithm CHARM

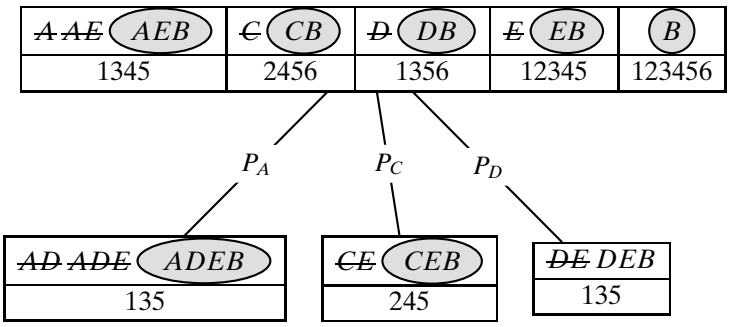
```
// Initial Call: C ← ∅, P ← {⟨i, t(i)⟩ : i ∈ I, sup(i) ≥ minsup}
CHARM (P, minsup, C):
1  Sort P in increasing order of support (i.e., by increasing |t(X_i)|)
2  foreach ⟨X_i, t(X_i)⟩ ∈ P do
3      P_i ← ∅
4      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
5          X_ij = X_i ∪ X_j
6          t(X_ij) = t(X_i) ∩ t(X_j)
7          if sup(X_ij) ≥ minsup then
8              if t(X_i) = t(X_j) then // Property 1
9                  Replace X_i with X_ij in P and P_i
10                 Remove ⟨X_j, t(X_j)⟩ from P
11             else
12                 if t(X_i) ⊂ t(X_j) then // Property 2
13                     Replace X_i with X_ij in P and P_i
14                 else // Property 3
15                     P_i ← P_i ∪ {⟨X_ij, t(X_ij)⟩}
16     if P_i ≠ ∅ then CHARM (P_i, minsup, C)
17     if ∄Z ∈ C, such that X_i ⊆ Z and t(X_i) = t(Z) then
18         C = C ∪ X_i    // Add X_i to closed set
```

it is pruned. AD is frequent and because $\mathbf{t}(A) \neq \mathbf{t}(D)$, we add $\langle AD, 135\rangle$ to the set P_A (property 3). When we combine A with E, property 2 applies, and we simply replace all occurrences of A in both P and P_A with AE, which is illustrated with the strike-through. Likewise, because $\mathbf{t}(A) \subset \mathbf{t}(B)$ all current occurrences of A, actually AE, in both P and P_A are replaced by AEB. The set P_A thus contains only one itemset $\{\langle ADEB, 135\rangle\}$. When CHARM is invoked with P_A as the IT-pair, it jumps straight to line 18, and adds $ADEB$ to the set of closed itemsets $\mathcal{C}$. When the call returns, we check whether AEB can be added as a closed itemset. AEB is a subset of $ADEB$, but it does not have the same support, thus AEB is also added to $\mathcal{C}$. At this point all closed itemsets containing A have been found.

The Charm algorithm proceeds with the remaining branches as shown in Figure 9.4b. For instance, C is processed next. CD is infrequent and thus pruned. CE is frequent and it is added to P_C as a new extension (via property 3). Because $\mathbf{t}(C) \subset \mathbf{t}(B)$, all occurrences of C are replaced by CB, and $P_C = \{\langle CEB, 245\rangle\}$. CEB and CB are both found to be closed. The computation proceeds in this manner until all closed frequent itemsets are enumerated. Note that when we get to DEB and perform the closure check, we find that it is a subset of $ADEB$ and also has the same support; thus DEB is not closed.



(a) Process A



(b) Charm

Figure 9.4. Mining closed frequent itemsets. Closed itemsets are shown as shaded ovals. Strike-through represents itemsets X_i replaced by $X_i \cup X_j$ during execution of the algorithm. Infrequent itemsets are not shown.

9.4 NONDERIVABLE ITEMSETS

An itemset is called *nonderivable* if its support cannot be deduced from the supports of its subsets. The set of all frequent nonderivable itemsets is a summary or condensed representation of the set of all frequent itemsets. Further, it is lossless with respect to support, that is, the exact support of all other frequent itemsets can be deduced from it.

Generalized Itemsets

Let $\mathcal{T}$ be a set of tids, let $\mathcal{I}$ be a set of items, and let X be a k-itemset, that is, $X = \{x_1, x_2, \ldots, x_k\}$. Consider the tidsets $\mathbf{t}(x_i)$ for each item $x_i \in X$. These k-tidsets induce a partitioning of the set of all tids into 2^k regions, some of which may be empty, where each partition contains the tids for some subset of items $Y \subseteq X$, but for none of the remaining items $Z = Y \setminus X$. Each such region is therefore the tidset of a *generalized itemset* comprising items in X or their negations. As such a generalized itemset can be represented as $Y\overline{Z}$, where Y consists of regular items and Z consists of negated items. We define the support of a generalized itemset $Y\overline{Z}$ as the number of transactions that

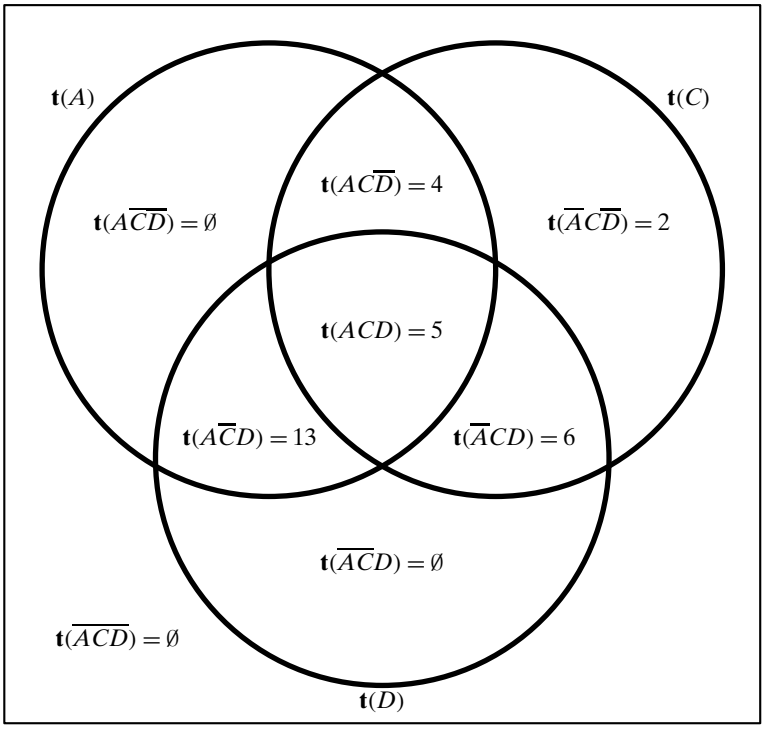


Figure 9.5. Tidset partitioning induced by $\mathbf{t}(A)$, $\mathbf{t}(C)$, and $\mathbf{t}(D)$.

contain all items in Y but no item in Z:

$$sup(Y\overline{Z}) = \left|\{t \in \mathcal{T} \mid Y \subseteq \mathbf{i}(t) \text{ and } Z \cap \mathbf{i}(t) = \emptyset\}\right|$$

Example 9.5. Consider the example dataset in Figure 9.1a. Let $X = ACD$. We have $\mathbf{t}(A) = 1345$, $\mathbf{t}(C) = 2456$, and $\mathbf{t}(D) = 1356$. These three tidsets induce a partitioning on the space of all tids, as illustrated in the Venn diagram shown in Figure 9.5. For example, the region labeled $\mathbf{t}(AC\overline{D}) = 4$ represents those tids that contain A and C but not D. Thus, the support of the generalized itemset $AC\overline{D}$ is 1. The tids that belong to all the eight regions are shown. Some regions are empty, which means that the support of the corresponding generalized itemset is 0.

Inclusion–Exclusion Principle

Let $Y\overline{Z}$ be a generalized itemset, and let $X = Y \cup Z = YZ$. The inclusion–exclusion principle allows one to directly compute the support of $Y\overline{Z}$ as a combination of the supports for all itemsets W, such that $Y \subseteq W \subseteq X$:

$$sup(Y\overline{Z}) = \sum_{Y \subseteq W \subseteq X} -1^{|W \setminus Y|} \cdot sup(W) \tag{9.2}$$

Example 9.6. Let us compute the support of the generalized itemset $\overline{A}C\overline{D} = C\overline{AD}$, where $Y = C$, $Z = AD$ and $X = YZ = ACD$. In the Venn diagram shown in Figure 9.5, we start with all the tids in $\mathbf{t}(C)$, and remove the tids contained in $\mathbf{t}(AC)$ and $\mathbf{t}(CD)$. However, we realize that in terms of support this removes $sup(ACD)$ twice, so we need to add it back. In other words, the support of $C\overline{AD}$ is given as

$$\begin{aligned} sup(C\overline{AD}) &= sup(C) - sup(AC) - sup(CD) + sup(ACD) \\ &= 4 - 2 - 2 + 1 = 1 \end{aligned}$$

But, this is precisely what the inclusion–exclusion formula gives:

$$\begin{aligned} sup(C\overline{AD}) = {} & (-1)^0\, sup(C) + && W = C, |W \setminus Y| = 0 \\ & (-1)^1\, sup(AC) + && W = AC, |W \setminus Y| = 1 \\ & (-1)^1\, sup(CD) + && W = CD, |W \setminus Y| = 1 \\ & (-1)^2\, sup(ACD) && W = ACD, |W \setminus Y| = 2 \\ = {} & sup(C) - sup(AC) - sup(CD) + sup(ACD) \end{aligned}$$

We can see that the support of $C\overline{AD}$ is a combination of the support values over all itemsets W such that $C \subseteq W \subseteq ACD$.

Support Bounds for an Itemset

Notice that the inclusion–exclusion formula in Eq. (9.2) for the support of $Y\overline{Z}$ has terms for all subsets between Y and $X = YZ$. Put differently, for a given k-itemset X, there are 2^k generalized itemsets of the form $Y\overline{Z}$, with $Y \subseteq X$ and $Z = X \setminus Y$, and each such generalized itemset has a term for $sup(X)$ in the inclusion–exclusion equation; this happens when $W = X$. Because the support of any (generalized) itemset must be non-negative, we can derive a bound on the support of X from each of the 2^k generalized itemsets by setting $sup(Y\overline{Z}) \geq 0$. However, note that whenever $|X \setminus Y|$ is even, the coefficient of $sup(X)$ is $+1$, but when $|X \setminus Y|$ is odd, the coefficient of $sup(X)$ is -1 in Eq. (9.2). Thus, from the 2^k possible subsets $Y \subseteq X$, we derive 2^{k-1} lower bounds and 2^{k-1} upper bounds for $sup(X)$, obtained after setting $sup(Y\overline{Z}) \geq 0$, and rearranging the terms in the inclusion–exclusion formula, so that $sup(X)$ is on the left hand side and the the remaining terms are on the right hand side

$$\textbf{Upper Bounds } (|X \setminus Y| \text{ is odd}): \qquad sup(X) \leq \sum_{Y \subseteq W \subset X} -1^{(|X \setminus Y|+1)} sup(W) \tag{9.3}$$

$$\textbf{Lower Bounds } (|X \setminus Y| \text{ is even}): \qquad sup(X) \geq \sum_{Y \subseteq W \subset X} -1^{(|X \setminus Y|+1)} sup(W) \tag{9.4}$$

Note that the only difference in the two equations is the inequality, which depends on the starting subset Y.

Example 9.7. Consider Figure 9.5, which shows the partitioning induced by the tidsets of A, C, and D. We wish to determine the support bounds for $X = ACD$ using each of the generalized itemsets $Y\overline{Z}$ where $Y \subseteq X$. For example, if $Y = C$, then the inclusion-exclusion principle [Eq. (9.2)] gives us

$$sup(C\overline{AD}) = sup(C) - sup(AC) - sup(CD) + sup(ACD)$$

Setting $sup(C\overline{AD}) \geq 0$, and rearranging the terms, we obtain

$$sup(ACD) \geq -sup(C) + sup(AC) + sup(CD)$$

which is precisely the expression from the lower-bound formula in Eq. (9.4) because $|X \setminus Y| = |ACD - C| = |AD| = 2$ is even.

As another example, let $Y = \emptyset$. Setting $sup(\overline{ACD}) \geq 0$, we have

$$\begin{aligned} sup(\overline{ACD}) &= sup(\emptyset) - sup(A) - sup(C) - sup(D) + \\ &\quad sup(AC) + sup(AD) + sup(CD) - sup(ACD) \geq 0 \\ \implies sup(ACD) &\leq sup(\emptyset) - sup(A) - sup(C) - sup(D) + \\ &\quad sup(AC) + sup(AD) + sup(CD) \end{aligned}$$

Notice that this rule gives an upper bound on the support of ACD, which also follows from Eq. (9.3) because $|X \setminus Y| = 3$ is odd.

In fact, from each of the regions in Figure 9.5, we get one bound, and out of the eight possible regions, exactly four give upper bounds and the other four give lower bounds for the support of ACD:

$$\begin{array}{lll} sup(ACD) & \geq 0 & \text{when } Y = ACD \\ & \leq sup(AC) & \text{when } Y = AC \\ & \leq sup(AD) & \text{when } Y = AD \\ & \leq sup(CD) & \text{when } Y = CD \\ & \geq sup(AC) + sup(AD) - sup(A) & \text{when } Y = A \\ & \geq sup(AC) + sup(CD) - sup(C) & \text{when } Y = C \\ & \geq sup(AD) + sup(CD) - sup(D) & \text{when } Y = D \\ & \leq sup(AC) + sup(AD) + sup(CD) - & \\ & \quad sup(A) - sup(C) - sup(D) + sup(\emptyset) & \text{when } Y = \emptyset \end{array}$$

This derivation of the bounds is schematically summarized in Figure 9.6. For instance, at level 2 the inequality is $\geq$, which implies that if Y is any itemset at this level, we will obtain a lower bound. The signs at different levels indicate the coefficient of the corresponding itemset in the upper or lower bound computations via Eq. (9.3) and Eq. (9.4). Finally, the subset lattice shows which intermediate terms W have to be considered in the summation. For instance, if $Y = A$, then the intermediate terms are $W \in \{AC, AD, A\}$, with the corresponding signs $\{+1, +1, -1\}$, so that we obtain the lower bound rule:

$$sup(ACD) \geq sup(AC) + sup(AD) - sup(A)$$

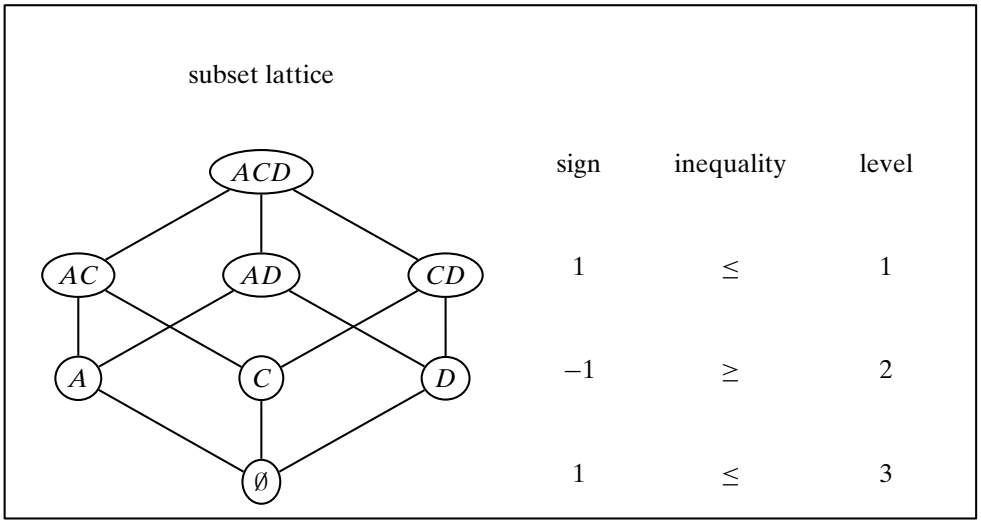


Figure 9.6. Support bounds from subsets.

Nonderivable Itemsets

Given an itemset X, and $Y \subseteq X$, let $IE(Y)$ denote the summation

$$IE(Y) = \sum_{Y \subseteq W \subset X} -1^{(|X \setminus Y|+1)} \cdot sup(W)$$

Then, the sets of all upper and lower bounds for $sup(X)$ are given as

$$UB(X) = \left\{ IE(Y) \middle| \; Y \subseteq X, \; |X \setminus Y| \text{ is odd} \right\}$$
$$LB(X) = \left\{ IE(Y) \middle| \; Y \subseteq X, \; |X \setminus Y| \text{ is even} \right\}$$

An itemset X is called *nonderivable* if $\max\{LB(X)\} \neq \min\{UB(X)\}$, which implies that the support of X cannot be derived from the support values of its subsets; we know only the range of possible values, that is,

$$sup(X) \in \Big[\max\{LB(X)\}, \min\{UB(X)\} \Big]$$

On the other hand, X is derivable if $sup(X) = \max\{LB(X)\} = \min\{UB(X)\}$ because in this case $sup(X)$ can be derived exactly using the supports of its subsets. Thus, the set of all frequent nonderivable itemsets is given as

$$\mathcal{N} = \left\{ X \in \mathcal{F} \mid \max\{LB(X)\} \neq \min\{UB(X)\} \right\}$$

where $\mathcal{F}$ is the set of all frequent itemsets.

Example 9.8. Consider the set of upper bound and lower bound formulas for $sup(ACD)$ outlined in Example 9.7. Using the tidset information in Figure 9.5, the

support lower bounds are

$$\begin{aligned} sup(ACD) &\geq 0 \\ &\geq sup(AC) + sup(AD) - sup(A) = 2 + 3 - 4 = 1 \\ &\geq sup(AC) + sup(CD) - sup(C) = 2 + 2 - 4 = 0 \\ &\geq sup(AD) + sup(CD) - sup(D) = 3 + 2 - 4 = 0 \end{aligned}$$

and the upper bounds are

$$\begin{aligned} sup(ACD) &\leq sup(AC) = 2 \\ &\leq sup(AD) = 3 \\ &\leq sup(CD) = 2 \\ &\leq sup(AC) + sup(AD) + sup(CD) - sup(A) - sup(C) - \\ &\quad sup(D) + sup(\emptyset) = 2 + 3 + 2 - 4 - 4 - 4 + 6 = 1 \end{aligned}$$

Thus, we have

$$\begin{aligned} LB(ACD) &= \{0, 1\} & \max\{LB(ACD)\} &= 1 \\ UB(ACD) &= \{1, 2, 3\} & \min\{UB(ACD)\} &= 1 \end{aligned}$$

Because $\max\{LB(ACD)\} = \min\{UB(ACD)\}$ we conclude that ACD is derivable.

Note that is it not essential to derive all the upper and lower bounds before one can conclude whether an itemset is derivable. For example, let $X = ABDE$. Considering its immediate subsets, we can obtain the following upper bound values:

$$\begin{aligned} sup(ABDE) &\leq sup(ABD) = 3 \\ &\leq sup(ABE) = 4 \\ &\leq sup(ADE) = 3 \\ &\leq sup(BDE) = 3 \end{aligned}$$

From these upper bounds, we know for sure that $sup(ABDE) \leq 3$. Now, let us consider the lower bound derived from $Y = AB$:

$$sup(ABDE) \geq sup(ABD) + sup(ABE) - sup(AB) = 3 + 4 - 4 = 3$$

At this point we know that $sup(ABDE) \geq 3$, so without processing any further bounds, we can conclude that $sup(ABDE) \in [3, 3]$, which means that $ABDE$ is derivable.

For the example database in Figure 9.1a, the set of all frequent nonderivable itemsets, along with their support bounds, is

$$\begin{aligned} \mathcal{N} = \big\{ & A[0,6], B[0,6], C[0,6], D[0,6], E[0,6], \\ & AD[2,4], AE[3,4], CE[3,4], DE[3,4] \big\} \end{aligned}$$

Notice that single items are always nonderivable by definition.

9.5 FURTHER READING

The concept of closed itemsets is based on the elegant lattice theoretic framework of formal concept analysis (Ganter, Wille, and Franzke, 1997). The Charm algorithm for mining frequent closed itemsets appears in Zaki and Hsiao (2005), and the GenMax method for mining maximal frequent itemsets is described in Gouda and Zaki (2005). For an Apriori style algorithm for maximal patterns, called MaxMiner, that uses very effective support lower bound based itemset pruning see Bayardo (1998). The notion of minimal generators was proposed in Bastide et al. (2000); they refer to them as *key patterns*. Nonderivable itemset mining task was introduced in Calders and Goethals (2007).

Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., and Lakhal, L. (2000). "Mining frequent patterns with counting inference." *ACM SIGKDD Explorations*, 2 (2): 66–75.

Bayardo R. J., Jr. (1998). "Efficiently mining long patterns from databases." *In Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, pp. 85–93.

Calders, T. and Goethals, B. (2007). "Non-derivable itemset mining." *Data Mining and Knowledge Discovery*, 14 (1): 171–206.

Ganter, B., Wille, R., and Franzke, C. (1997). *Formal Concept Analysis: Mathematical Foundations*. New York: Springer-Verlag.

Gouda, K. and Zaki, M. J. (2005). "Genmax: An efficient algorithm for mining maximal frequent itemsets." *Data Mining and Knowledge Discovery*, 11 (3): 223–242.

Zaki, M. J. and Hsiao, C.-J. (2005). "Efficient algorithms for mining closed itemsets and their lattice structure." *IEEE Transactions on Knowledge and Data Engineering*, 17 (4): 462–478.

9.6 EXERCISES

Q1. True or False:

- **(a)** Maximal frequent itemsets are sufficient to determine all frequent itemsets with their supports.
- **(b)** An itemset and its closure share the same set of transactions.
- **(c)** The set of all maximal frequent sets is a subset of the set of all closed frequent itemsets.
- **(d)** The set of all maximal frequent sets is the set of longest possible frequent itemsets.

Q2. Given the database in Table 9.1

- **(a)** Show the application of the closure operator on AE, that is, compute $\mathbf{c}(AE)$. Is AE closed?
- **(b)** Find all frequent, closed, and maximal itemsets using $minsup = 2/6$.

Q3. Given the database in Table 9.2, find all minimal generators using $minsup = 1$.

Table 9.1. Dataset for Q2

Tid	Itemset
t_1	ACD
t_2	BCE
t_3	$ABCE$
t_4	BDE
t_5	$ABCE$
t_6	$ABCD$

Table 9.2. Dataset for Q3

Tid	Itemset
1	ACD
2	BCD
3	AC
4	ABD
5	$ABCD$
6	BCD

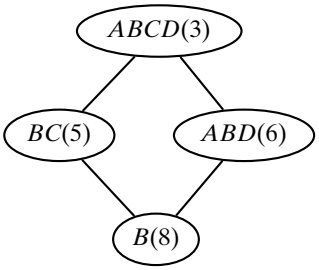


Figure 9.7. Closed itemset lattice for Q4.

Q4. Consider the frequent closed itemset lattice shown in Figure 9.7. Assume that the item space is $\mathcal{I} = \{A, B, C, D, E\}$. Answer the following questions:

(a) What is the frequency of CD?

(b) Find all frequent itemsets and their frequency, for itemsets in the subset interval $[B, ABD]$.

(c) Is ADE frequent? If yes, show its support. If not, why?

Q5. Let $\mathcal{C}$ be the set of all closed frequent itemsets and $\mathcal{M}$ the set of all maximal frequent itemsets for some database. Prove that $\mathcal{M} \subseteq \mathcal{C}$.

Q6. Prove that the closure operator $\mathbf{c} = \mathbf{i} \circ \mathbf{t}$ satisfies the following properties (X and Y are some itemsets):

(a) Extensive: $X \subseteq \mathbf{c}(X)$

(b) Monotonic: If $X \subseteq Y$ then $\mathbf{c}(X) \subseteq \mathbf{c}(Y)$

(c) Idempotent: $\mathbf{c}(X) = \mathbf{c}(\mathbf{c}(X))$

Table 9.3. Dataset for Q7

Tid	Itemset
1	ACD
2	BCD
3	ACD
4	ABD
5	$ABCD$
6	BC

Q7. Let δ be an integer. An itemset X is called a δ-*free* itemset iff for all subsets $Y \subset X$, we have $sup(Y) - sup(X) > \delta$. For any itemset X, we define the δ-*closure* of X as follows:

$$\delta\text{-closure}(X) = \{Y \mid X \subset Y, sup(X) - sup(Y) \leq \delta, \text{ and } Y \text{ is maximal}\}$$

Consider the database shown in Table 9.3. Answer the following questions:

(a) Given $\delta = 1$, compute all the δ-free itemsets.

(b) For each of the δ-free itemsets, compute its δ-closure for $\delta = 1$.

Q8. Given the lattice of frequent itemsets (along with their supports) shown in Figure 9.8, answer the following questions:

(a) List all the closed itemsets.

(b) Is BCD derivable? What about $ABCD$? What are the bounds on their supports.

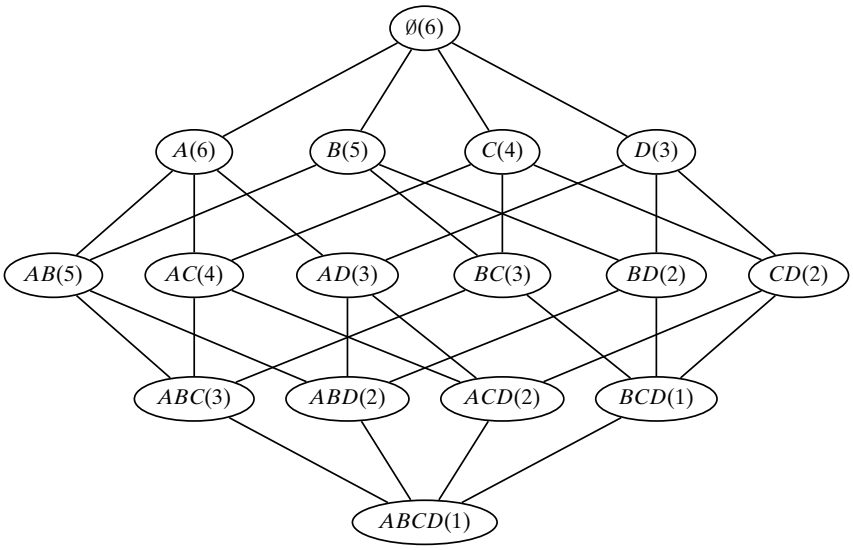


Figure 9.8. Frequent itemset lattice for Q8.

Q9. Prove that if an itemset X is derivable, then so is any superset $Y \supset X$. Using this observation describe an algorithm to mine all nonderivable itemsets.