Association Rules Mining

Reference

- R. Agrawal, T. Imielinski, and A. Swami. **Mining association rules between sets of items in large databases**. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington D.C., May 1993
- R. Agrawal and R. Srikant. ***Fast algorithms for mining association rules in large databases***. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994

Table of Contents

Introduction

- Retail organizations have a massive amounts of sales data, referred to as the basket data.
  - From that data, how can we know the associations between items bought in a transaction?
- In 1993, Rakes Agrawal proposed a method called: Association Rules Mining.

Introduction

Transaction ID	Items bought
1	milk, butter, juice, coke, bread
2	bread, rice, butter, milk, pasta
3	potatoes, butter, milk, bread
…	…
1988	Pepsi, bread, milk, butter, sausage

- An association rule is:

[Butter, Bread] $\Rightarrow$ [Milk]

Formal statement & Decomposition

- Set of items: $\mathbf{I} = \{i_1, i_2, i_3, \ldots, i_n\}$
- Transaction: $(TransacId, T) : T \subset \mathbf{I}$
- Itemset, k-itemset: $X, Y \subset \mathbf{I}$
- Association rule: $X \Rightarrow Y : X \cap Y = \Phi$
- Support
  - Support of the itemset X: Number of transactions that contain X
  - Support of the rule X => Y : Support of $X \cup Y$

Formal statement & Decomposition

- Confidence

$$Conf(X \Rightarrow Y) = \frac{Support(X \Rightarrow Y)}{Support(X)}$$

- Association rules mining: Generate all association rules that have support and confidence greater than the user-specified minimum support (minSup) and minimum confidence (minConf) respectively.

Formal statement & Decomposition

- Association Rules Mining can be decomposed into 2 sub problems:
- Sub problem 1: Finding all frequent itemsets.
  - X is a frequent itemset ⇔ Support(X) >= minSup
  - X is a frequent itemset ⇔ All subsets of X are frequent itemsets.

Formal statement & Decomposition

- Sub problem 2: Use the frequent itemsets to generate the desired rules.

$$\left.\begin{array}{l} Z \text{ is a frequent itemset} \\ X \subseteq Z, Y \subseteq Z \\ X \cap Y = \Phi \\ Conf(X \Rightarrow Y) \geq MinConf \end{array}\right\} \Rightarrow X \Rightarrow Y \text{ is an association rule}$$

  - X may be find by exhaustive search.

Outline

- Introduction
- Formal statement & Decomposition
- Apriori Algorithm
- Conclusion

Apriori Algorithm - Concepts

Itemsets: {I1, I2, I5}, {I2}, ...

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Item: I4

Transaction: T800

- X is frequent iff support(X) >= min_sup

Ex: min_sup = 3/9, =support{I1, I2} = 4/9. So {I1, I2} is frequent

- X is a k-itemset iff X has k items

Ex: {I1, I2, I5} is a 3-itemset

Apriori Algorithm - Purpose

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

- Find all k-itemsets having support >= min_sup
- k = 1, 2, …, 5

Apriori Algorithm – Step by Step

min_sup = 2/9
minimum support count = 2

Scan D for count of each candidate →

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Apriori Algorithm - Properties

- Use prior knowledge
- Apriori property: All nonempty subsets of a frequent itemset must also be frequent

- Iterative search for frequent itemsets
- Compute k+1-itemsets from k-itemsets

Apriori Algorithm - Methods

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

L_1 = find_frequent_1-itemsets(D);
for $(k = 2; L_{k-1} \neq \phi; k{+}{+})$ {
    C_k = apriori_gen(L_{k-1});
    for each transaction $t \in D$ { // scan D for counts
        C_t = subset(C_k, t); // get the subsets of t that are candidates
        for each candidate $c \in C_t$
            c.count++;
    }
    $L_k = \{c \in C_k | c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$;

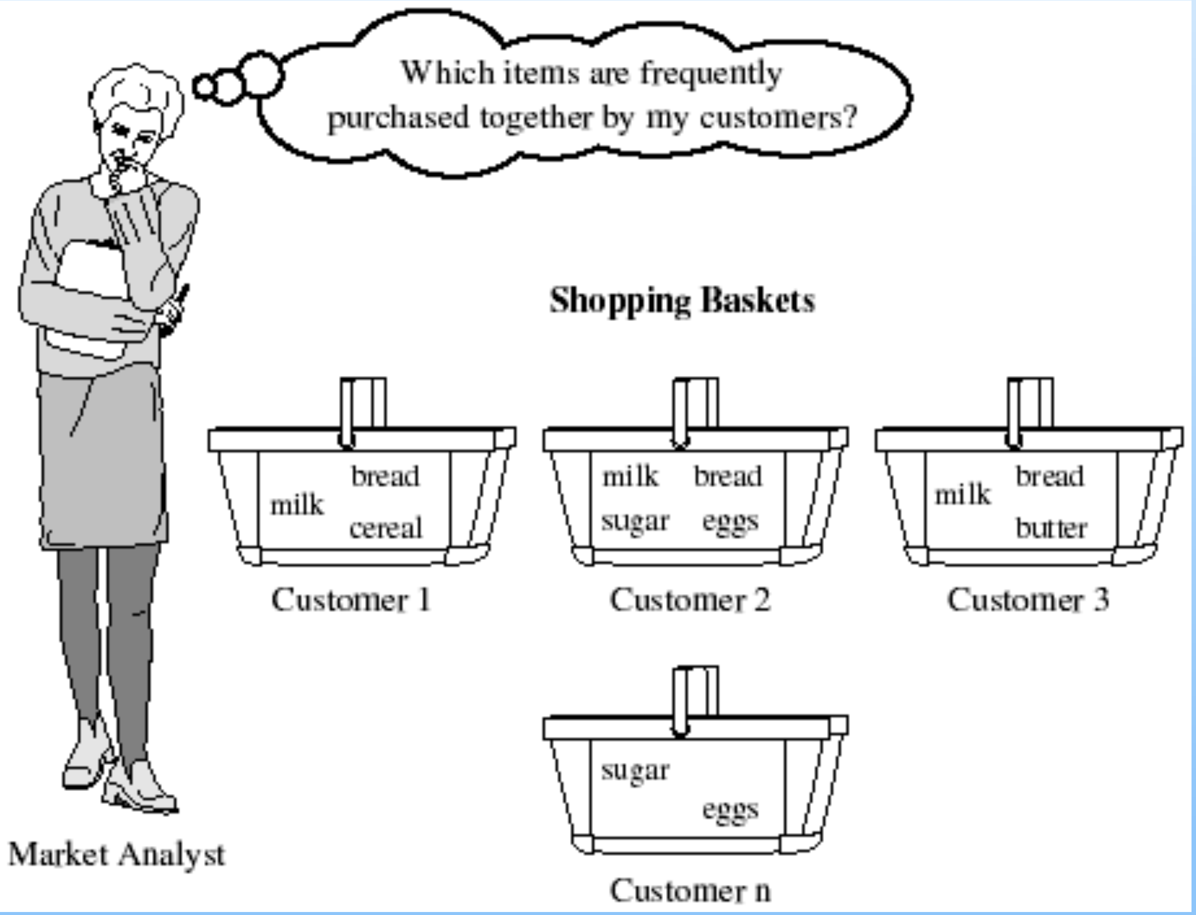
Apriori Algorithm - Methods

procedure apriori_gen(L_{k-1}:frequent $(k-1)$-itemsets)
for each itemset $l_1 \in L_{k-1}$
for each itemset $l_2 \in L_{k-1}$
if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ **then** {
$c = l_1 \bowtie l_2$; // join step: generate candidates
if has_infrequent_subset(c, L_{k-1}) **then**
delete c; // prune step: remove unfruitful candidate
else add c **to** C_k;
}
return C_k;

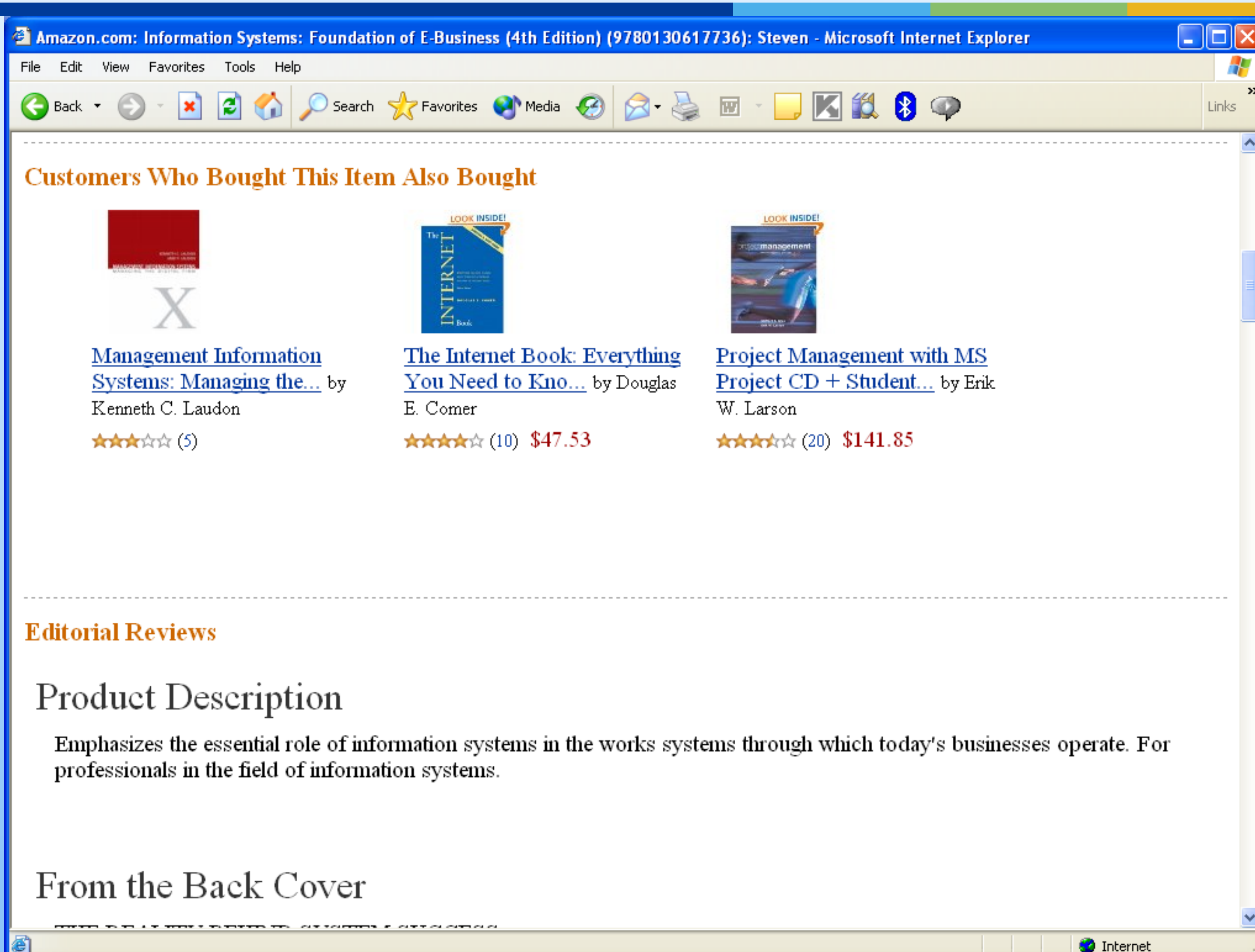
procedure has_infrequent_subset(c: candidate k-itemset;
L_{k-1}: frequent $(k-1)$-itemsets); // use prior knowledge
for each $(k-1)$-subset s **of** c
if $s \notin L_{k-1}$ **then**
return TRUE;
return FALSE;

Conclusion

- Case Study 1 – Market Basket Analysis



Case Study 2 – Related Products



Other applicatioins

- Catalog design
- Classification
- Clustering