Levi Sutton

CS 4412 (Tuesday/Thursday 6:30)

Homework 1 (Part 1)

|  |  |
| --- | --- |
| TID | Items\_brought |
| T100 | {M,O,N,K,E,Y} |
| T200 | {D,O,N,K,E,Y} |
| T300 | {M,A,K,E,} |
| T400 | {M,U,C,K,Y} |
| T500 | {C,O,O,K,I,E} |

Part a)

**Apriori Method:**

Part 1(initial itemset, Candidate generation denoted by Cd subscript k, k = 1,2, 3, so on):

Cd1:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| {A} | 1 |
| {C} | 2 |
| {D} | 1 |
| {E} | 4 |
| {I} | 1 |
| {K} | 5 |
| {M} | 3 |
| {N} | 2 |
| {O} | 3 (repeated in T500) |
| {U} | 1 |
| {Y} | 3 |

Red == Eliminated

Part 2 (Frequent letters in itemset):

Ld1:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| {K} | 5 |
| {E} | 4 |
| {M} | 3 |
| {O} | 3 |
| {Y} | 3 |

Part 3 (Itemset of pairs in Ld1):

Cd2:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| { E,K} | 4 |
| { E,M} | 2 |
| { E,O} | 3 |
| { E,Y} | 2 |
| { K,M} | 3 |
| { K,O} | 3 |
| { K,Y} | 3 |
| {M,O} | 1 |
| {M,Y} | 2 |
| {O,Y} | 2 |

Part 4 (Frequent letters in Cd2):

Ld2:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| { E,K} | 4 |
| { E,O} | 3 |
| { K,M} | 3 |
| { K,O} | 3 |
| { K,Y} | 3 |

Part 5 (Itemset of pairs in Ld2):

Cd3:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| { E,K,M} | 2 |
| { E,K,O} | 3 |
| { E,K,Y} | 2 |
| { K,O,Y} | 2 |
| { K,M,Y} | 2 |
| { M,O,Y} | 1 |
| { M,E,Y} | 1 |

Part 5 (Frequent letters in Cd3):

Ld3:

|  |  |
| --- | --- |
| Itemset | Number of occurrence |
| { E,K,O} | 3 |

**FP-Growth:**

FP-Growth vs Apriori Method

Apriori Method is an array based algorithm that uses the Join and Prune technique. It uses a breadth-first method and utilizes a level-wise approach where it generates patterns containing 1 item, 2, 3, etc. Candidate generation is slow and runtime increases exponentially depending on the number of different item sets. It requires a large memory space due to the growing amount of candidate generations. It also scans the database multiple times for generating candidate sets. FP-Growth is a tree based algorithm that constructs a conditional frequent pattern tree and pattern base. It uses a depth-first method and utilizes a pattern-growth approach that only considers pattern that exist in a database. Runtime increases linearly, depending on the number of transactions and items. Data is interdependent and each node needs the root. It requires less memory space and no candidate generation. It scans the database two time for constructing a frequent pattern tree.

Part b)

Count = 5

Support ({E,K,O} 🡪 items\_bought) = 3 / 5 = 0.6 = 60%

Confidence({E,K} 🡪 {O}) = 3 / 4 = 0.75 = 75% **\*(Not an association .75 < .80 (Min Confidence)\***

Confidence({E,O} 🡪 {K}) = 3 / 3 = 1 = 100%

Confidence({O,K} 🡪 {E}) = 3 / 3 = 1 = 100%

Min Confidence = 1 (Max confidence) / .60 (Support) = .80 = 80%

**Association Results:**

{E,O} 🡪 {K} (0.6, 1)

{O,K} 🡪 {E} (0.6, 1)