Frequent Itemset Mining of Distributed Uncertain Data under User-Defined Constraints

Alfredo Cuzzocrea
ICAR-CNR & University of Calabria, Italy

Carson K. Leung
University of Manitoba, Canada

SEBD 2012
Outline

• Introduction & related work

• Our proposed distributed mining system
  ▪ Finding constrained locally frequent itemsets
    • Step 1: Identification of items satisfying the constraint
    • Step 2: Construction of an UF-tree
    • Step 3: Mining of constrained frequent itemsets from the UF-tree
  ▪ Finding constrained globally frequent itemsets

• Experimental results

• Conclusions
Introduction

• **Frequent pattern mining (FPM)**
  - A data mining task
  - Non-trivial extraction of implicit, previously unknown, & potentially useful information—in the form of *frequently occurring collections of merchandise items or events*—from data

Cuzzocrea & Leung (SEBD 2012)
Related Work (1)

• **Apriori**
  - Generate-and-test paradigm

• **FP-growth**
  - Restricted test-only approach

• **UF-growth**
  - Mines frequent itemsets from uncertain data
  - Mines a centralized DB of uncertain data for all (unconstrained) frequent itemsets

Cuzzocrea & Leung (SEBD 2012)
Related Work (2)

• **DCF**
  - Mines constrained frequent itemsets from traditional precise data
  - Mines a centralized DB of precise data

• **FDM & Parallel-HFP-Leap**
  - Distributed mining
  - Do not handle constraints
  - Do not mine uncertain data
Our Proposed Distributed Mining System

• Non-trivial integration of
  ▪ constrained mining,
  ▪ distributed mining,
  ▪ uncertain data mining, with
  ▪ tree-based frequent itemset mining.

• Efficiently mines from distributed uncertain data for only those constrained frequent itemsets
Our Proposed Distributed Mining System

• Given:
  ▪ \( p \) sites/processors
  ▪ \( m = m_1 + m_2 + \ldots + m_p \) sensors in a distributed network
  ▪ \( m_1 \) wireless sensors transmit data to their closest or designated site/processor \( P_1 \)
  ▪ \( m_2 \) sensors transmit data to the site/processor \( P_2 \)
  ▪ etc.

• finds
  a) constrained itemsets that are locally frequent w.r.t. site/processor \( P_i \) and
  b) those that are globally frequent w.r.t. all sites/processors in the entire wireless sensor network
A. Finding Constrained Locally Frequent Itemsets

• Step 1:
  - Identification of items satisfying the constraint

• Step 2:
  - Construction of an UF-tree

• Step 3:
  - Mining of constrained frequent itemsets from the UF-tree
A1. Identification of Items Satisfying the SAM Constraint

• Succinct anti-monotone (SAM) constraint
  ▪ Any X satisfying $C_{SAM}$ must be generated by combining items from ItemM
    • Items in ItemM can be efficiently enumerated (from the list of domain items) by selecting only those items that individually satisfy $C_{SAM}$
  ▪ An itemset X satisfying $C_{SAM}$ cannot contain any item from ItemO
    • E.g., if an itemset X containing an item having price > $25, then X violates $C_{SAM}$ & so does every superset of X
A1. Identification of Items Satisfying the SUC Constraint

- Succinct non-anti-monotone (SUC) constraint
  - Any itemset $X$ satisfying $C_{\text{SUC}}$ must be generated by combining at least one ItemM item and possibly some ItemO items
  - If $X$ violates $C_{\text{SUC}}$, there is no guarantee that all or any of its supersets would violate $C_{\text{SUC}}$
  - Any itemset $X$ satisfying $C_{\text{SUC}}$ is composed of mandatory items (i.e., items that individually satisfy $C_{\text{SUC}}$) and possibly some optional items (regardless whether or not they satisfy $C_{\text{SUC}}$)
A2. Construction of an UF-Tree (1)

• Classifies domain items into $\text{ItemM}$ & $\text{ItemO}$ items
  ▪ No $\text{ItemO}$ items for $C_{\text{SAM}}$

• Constructs an UF-tree
  ▪ Scans the DB of uncertain data once
  ▪ Accumulates the expected support of each of the items
  ▪ Discards infrequent items
  ▪ Only captures frequent items in the UF-tree
    • Any infrequent $\text{ItemM}$ or $\text{ItemO}$ items can be safely discarded because any itemset containing an infrequent item is also infrequent
A2. Construction of an UF-Tree (2)

- Arranges \textbf{ItemM} items to appear below \textbf{ItemO} items
  - \textbf{ItemM} items are closer to the leaves
  - \textbf{ItemO} items are closer to the root
- Sorts all the items \textbf{ItemM} in non-ascending order of accumulated expected support
- Sorts all the items \textbf{ItemO} in non-ascending order of accumulated expected support
A2. Construction of an UF-Tree (3)

- Scans the DB the second time; inserts each transaction of the DB into the UF-tree
  - New transaction is merged with a child (or descendant) node of the root of the UF-tree (at the highest support level) only if the same item & the same expected support exist in both the transaction & the child (or descendant) nodes
  - For $C_{SAM}$, UF-tree captures only those frequent ItemM items
A3. Mining of Constrained Frequent Itemsets from the UF-Tree

• Extracts appropriate paths to form a projected DB for each $x$ in $\text{ItemM}$
  - Does not need to form projected DBs for any $y$ in $\text{ItemO}$ because all itemsets satisfying $C_{SUC}$ must be “extensions” of an item from $\text{ItemM}$ (i.e., all valid itemsets must be grown from $\text{temM}$ items)
  - For $C_{SAM}$, no $\text{ItemO}$ items are kept in the UF-tree

• Recursively ...
  - constructs a UF-tree for each projected DB
  - mines all frequent itemsets that satisfy $C_{SAM}$ or $C_{SUC}$
B. Finding Constrained Globally Frequent Itemsets (1)

• Each site/processor $P_i$ (for $1 \leq i \leq p$)
  ▪ applies constraint checking & frequency checking to find locally frequent $\text{ItemM}_i$ items (& $\text{ItemO}_i$ items for $C_{\text{SUC}}$)
  ▪ transmits locally frequent $\text{ItemM}_i$ items (& $\text{ItemO}_i$ items for $C_{\text{SUC}}$) to a centralized site/processor $Q$

• Centralized site/processor $Q$
  ▪ takes the union of these items
  ▪ broadcasts the union to all $P_i$'s
B. Finding Constrained Globally Frequent Itemsets (2)

• Each $P_i$
  ▪ extracts these potentially globally frequent items from transactions in $TDB_i$ & puts into an UF-tree
  ▪ UF-tree contains ...
    • items that are locally frequent w.r.t. $P_i$
    • items that are potentially globally frequent but locally infrequent items w.r.t $P_i$
  ▪ recursively applies the usual tree-based mining process to each $\alpha$-projected DB (where locally frequent $\alpha \subseteq \text{ItemM}_i$) of the UF-tree at $P_i$ to find constrained locally frequent itemsets (with local frequency info) & send these itemsets to Q (where the local frequencies are summed)

• If the sum of available local frequencies of a constrained itemset $X \geq$ minimum support threshold, then $X$ is *globally frequent*

• For the case where a constrained itemset is locally frequent at a site $P_1$ but not at another site $P_2$, then Q sends a request to $P_2$ for finding its local frequency
Experimental Setup

- Datasets:
  - IBM synthetic data
  - Real-life DBs from ...
    - UC Irvine Machine Learning Depository
    - Frequent Itemset Mining Implementation (FIMI) Dataset Repository
Experimental Results (1)

• Accuracy
  ▪ As accurate as UF-growth
    (and they both returned the same collection of frequent itemsets)

• Flexibility
  ▪ More flexible than UF-growth
    • Our system is capable of finding frequent itemsets from distributed uncertain data with constraints of any selectivity
    • UF-growth is confined to those of 100% selectivity
Experimental Results (2)

• Effectiveness of constrained mining in a distributed environment
  ▪ When selectivity of constraints decreased,
    • amount of communication/data transmitted between the distributed sites $P_i$ & their centralized site $Q$ decreased
    • runtimes decreased
Experimental Results (3)

• Effects of varying #distributed sites
  ▪ When more sites were in the distributed network,
    • transmitted more data
      » because an addition of a site implies transmission of an additional set of locally frequent items and locally frequent itemsets
    • runtime increased slightly
      » because the extra communication time was offset by the savings in building and mining from a smaller UF-tree at each site
Experimental Results (4)

(a) Runtime vs. selectivity
(b) Runtime vs. minsup
(c) Runtime vs. existential probability
(d) Amt of transmitted data vs. selectivity
Thank you  Grazie

• More info, please refer to our paper

  or

  contact us:

  dblab@cs.umanitoba.ca
  cuzzocrea@si.deis.unical.it