Sequential Data

COMP 527 Data Mining
Danushka Bollegala

UNIVERSITY OF LIVERPOOL

Types of Sequential Data

- Natural Language Texts
  - Lexical or POS patterns that represent semantic relations between entities
  - *Tim Cook* is the CEO of *Apple*
  - **X** is the CEO of **Y**
- Microarray Data (gene expression data)
- Substrings, subsequences in a database
  - frequent item sets
- Time series analysis
  - Stock market prediction
  - Foreign exchange rate prediction

Sequential Pattern Mining

- The Problem
  - Given a database D, which stores strings consisting letters from some alphabet, find all substrings (or subsequences) in D that occur some pre-defined minimum value.

Substrings vs. Subsequences

- Substring of a string must have *consecutive* elements selected from the substring
  - note, ote, book, ebook are all substrings of the word notebook
- Subsequence of a string can have gaps (must be in the same relative order but need not be consecutive)
  - {n,t,k}, {n,o,b,k}, {n,o,o,k} are all subsequences of the word notebook

Quiz

- Give a word *w* of n unique letters (the maximum number of occurrence of any letter within the word is 1)
  - How many unique substrings can you generate from *w*?
  - How many unique subsequences can you generate from *w*?

Naive Algorithm

- Take each string *s* in the given database *D*
    - Generate all substrings/subsequences of *s*
- Count the occurrences of each subsequence generated in the previous step
- Select the substrings/subsequences that occur more than the pre-defined threshold

Problems with the Naive Algorithm

- The number of all substrings/subsequences can be very large in practice
  - storing all substrings will be challenging because of the memory requirements
  - counting the occurrences mean we need to update a hash table (eg. Python dictionary) with the substrings/subsequences as the keys
    - Time consuming
- Waste of resources and computation because we will be deleting most of the generated and stored substrings/ subsequences in the end.

Apriori Algorithm

- An efficient algorithm for extracting frequent itemsets from a database
- One of the most famous data mining algorithms
  - Proposed by Rakesh Agrawal@MSR in 1994
- *Fast Algorithms for Mining Association Rules*, R. Agrawal and R. Srikant, VLDB, 1994
- Apriori
  - Following from theoretical deduction rather than from observation or experience.

Terminology (1/3)

- Item
  - A product, entity, letter (any object for that matter) stored in a database
- Itemset
  - A set of items
- Transaction
  - A record in a database
- Example
  - John bought milk, sugar, and eggs when he went to the supermarket last Sunday.
  - Itemset = {milk, sugar, eggs}
  - Individual items = milk, sugar, eggs
  - Transaction: Because all three items above were purchased at the same time it can be considered as a single transaction.
- We can assign a transaction id (TID) for a transaction for the ease of reference
  - eg. 001, 002, etc.

Terminology (2/3)

- Support (frequency) of an itemset
  - The number of different transactions in which a particular itemset appears as a subset is defined as the frequency of that itemset
  - If {milk, sugar} appears in 10 different transactions in the database, then 10 is the frequency of the itemset {milk, sugar}.
- The length of an itemset
  - The number of items in an itemset is defined as its length.
  - The length of {milk, sugar} is 2.

Terminology (3/3)

- minimum support
  - As we discussed previously, there can be a large number of subsets (itemsets) in a database.
  - Often we are interested in extracting itemsets that occur more than in a minimum number of transactions (with a minimum support)
- Large itemsets
  - Itemsets that have support larger than the minimum support are said to be *large itemsets*.
- Small itemsets
  - Itemsets that have support smaller than the minimum support are said to be *smaller itemsets.*
- large and small in the above definitions *does not* relate to the length of the itemset!

Apriori Property

- If S is a large itemset (with minimum support *t*), then any subset of S is also a large itemset (with minimum support *t*)
- Obvious???
    - This property, although obvious when someone mentions it, can be used to reduce the search space significantly, thereby speeding up the discovery of large itemsets

Apriori Algorithm

L_1 = {large 1-itemsets}
for ($k=2$; $L_{k-1} \neq \varnothing$; k++) do
  $C_k = gen(L_{k-1})$
  forall transactions $t \in D$ do
    C_t = subset(C_k, t)
    forall candidates $c \in C_t$ do
      c.count++
  end
  $L_k = \{c \in C_k \mid c.count \geq minsup\}$
end
return $\bigcup_k L_k$

<u>Notation</u>

L_k length k itemset

C_k candidate set of length k

minsup: minimum support

$\varnothing$ null set

gen function (1/2)

- To generate the candidates of length k, we compute the *join* of all itemsets in L_{k-1}
  - We find two itemsets *p* and *q* that match upto length (k-2) and differ only at the (k-1)-th item and create an itemset of length k by including the matching component, (k-1)-th item of *p* and the (k-1)-th item of *q*.

$$
\begin{array}{l}
\textbf{insert into } C_k \\
\textbf{select } p.\mathrm{item}_1,\ p.\mathrm{item}_2,\ \ldots,\ p.\mathrm{item}_{k-1},\ q.\mathrm{item}_{k-1} \\
\textbf{from } L_{k-1}\ p,\ L_{k-1}\ q \\
\textbf{where } p.\mathrm{item}_1 = q.\mathrm{item}_1,\ \ldots,\ p.\mathrm{item}_{k-2} = q.\mathrm{item}_{k-2}, \\
\qquad p.\mathrm{item}_{k-1} < q.\mathrm{item}_{k-1};
\end{array}
$$

gen function (2/2)

- Filter out all candidates that are have (k-1) length subsets that are not in L_{k-1}

forall itemsets $c \in C_k$ **do**
    forall $(k-1)$-subsets s of c **do**
        if $(s \notin L_{k-1})$ **then**
            delete c from C_k;

Example

TID	Itemset
T001	{1,2}
T002	{2,3,4}
T003	{3,4}
T004	{1,3}
T005	{1,2,3,4}
T006	{2,4}

minimum frequency = 2
minimum support = 2/6 = 0.33
items in an itemset are ordered in the alphabetical order to speed up the subset search.

Quiz

Find all large itemsets of the given database.

L_1

itemset	frequency
1	$3 \geq 2$
2	$4 \geq 2$
3	$4 \geq 2$
4	$4 \geq 2$

$L_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$

$C_2 = \{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}\}$

L_2

Itemset	frequency
{1,2}	$2 \geq 2$
{1,3}	$2 \geq 2$
{1,4}	$1 \ngeq 2$
{2,3}	$2 \geq 2$
{2,4}	$3 \geq 2$
{3,4}	$3 \geq 2$

{1,4} is small.

Therefore,
L_2 = {{1,2}, {1,3,}, {2,3}, {2,4}, {3,4}}

C_3 = {{1,2,3}, {2,3,4}}

L_3

Itemset	frequency
{1,2,3}	1 ≱ 2
{2,3,4}	2 ≱2

$C_3 = \{\{1,2,3\}, \{2,3,4\}\}$

Therefore,
$L_3 = \{\{2,3,4\}\}$

This is the only itemset and we cannot generate candidates of length 4.
Thus, $c_4 = \varnothing$

Termination

- There are no itemsets for C_4
- Therefore the apriori algorithm terminates and returns the three itemsets
  - {1}, {2}, {3}, {4}, {1,2}, {1,3}, {2,3}, {2,4}, {3,4}, and {2,3,4} with minmum support 0.33 (2/6) for the given database of 6 transactions.