Web Data Mining

Syllabus

Data Mining Foundations: Association Rules and Sequential Patterns; Information Retrieval and Web Search: Information Retrieval Models, Relevance Feedback, Evaluation Measures, Text and Web Page Pre-Processing, Combining Multiple Rankings, Spamming; Social Network Analysis: Co-Citation and Bibliographic Coupling, PageRank, Hypertext Induced Topic Search, Community Discovery; Web Crawling: Basic Algorithm, Implementation Issues, Types; Structured Data Extraction: Wrapper Generation; Information Integration; Opinion Mining and Sentiment Analysis; Web Usage Mining; Building web scrapper; Writing web crawlers; Legalities and ethics of web scraping.

Books

- B. Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer, 2nd edition, 2011.
- R. Mitchell, Web scraping with Python: Collecting more data from the modern web, O'Reilly, 2nd edition, 2018.

Exams

- Midsem Exam (2 separate exams)
- EndSem (3 separate exams)
- No separate test
- No marks for attendance
- Assignments not decided

R1

Motivation

- Web contains a lot of data
- Manual collecting/processing this data is difficult

Slide 5

R1 Rohit, 28-12-2021

Chapter 2: Association Rules & Sequential Patterns

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Sequential pattern mining
- Summary

Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

The model: data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*.
- Transaction t :
  - t a set of items, and $t \subseteq I$.
- Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$.

Transaction data: supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}

  t2: {apple, eggs, salt, yogurt}

  … …

  tn: {biscuit, eggs, milk}

- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data: a set of documents

- **A text document data set. Each document is treated as a "bag" of keywords**

doc1:	Student, Teach, School
doc2:	Student, School
doc3:	Teach, School, City, Game
doc4:	Baseball, Basketball
doc5:	Basketball, Player, Spectator
doc6:	Baseball, Coach, Game, Team
doc7:	Basketball, Team, City, Game

The model: rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$.
- An association rule is an implication of the form:

  $$X \to Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset

Rule strength measures

- Support: The rule holds with support *sup* in *T* (the transaction data set) if sup% of transactions contain $X \cup Y$.
  - $sup = \Pr(X \cup Y)$.
- Confidence: The rule holds in *T* with confidence *conf* if *conf*% of tranactions that contain *X* also contain *Y*.
  - $conf = \Pr(Y \mid X)$
- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability.

Support and Confidence

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *T* has *n* transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - Mining with data on hard disk (not in memory)

An example

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

- Transaction data
- Assume:

  minsup = 30%
  minconf = 80%
- An example frequent *itemset*:

  {Chicken, Clothes, Milk}   [sup = 3/7]
- Association rules from the itemset:

  Clothes $\rightarrow$ Milk, Chicken   [sup = 3/7, conf = 3/3]

  …   …

  Clothes, Chicken $\rightarrow$ Milk,   [sup = 3/7, conf = 3/3]

Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered. E.g,
  - the quantity of each item purchased and
  - the price paid.

Many mining algorithms

- There are a large number of them!!
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
  - Given a transaction data set *T*, and a minimum support and a minimum confident, the set of association rules existing in *T* is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study only one: the Apriori Algorithm

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Sequential pattern mining
- Summary

The Apriori algorithm

- **The best known algorithm**
- **Two steps**:
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
  - Use frequent itemsets to generate rules.

- E.g., a frequent itemset

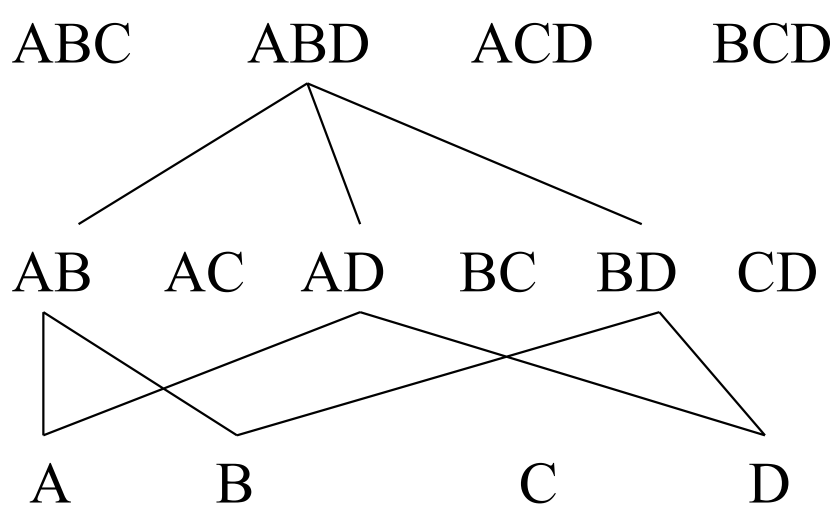
  {Chicken, Clothes, Milk} [sup = 3/7]

  and one rule from the frequent itemset

  Clothes $\rightarrow$ Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
  - In each iteration k, only consider itemsets that contain some k-1 frequent itemset.

- Find frequent itemsets of size 1: F_1
- From $k = 2$
  - C_k = candidates of size k: those itemsets of size k that could be frequent, given F_{k-1}
  - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).