Link Analysis II – Frequent Itemsets I

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
June 9, 2022

TODAY

Overview

- *Link Analysis II*
  - Link Spam and TrustRank: Fight Advanced Spammer Strategies
  - Hubs and Authorities: Alternative, Non-PageRank Approach
- *Frequent Itemsets I*
  - The Market-Basket Model
  - Frequent Itemsets: Definition and Applications
  - Association Rules
  - The A-Priori Algorithm

Learning Goals: Understand these topics and get familiarized

Link Spam

Link Spam: Introduction

- Google rendered *term spam ineffective*
- Spammers developed *link spam* as a technique to artificially increase PageRank
- In the following, understand how to
  - create link spam
  - and how to fight it

Spammer View of Web

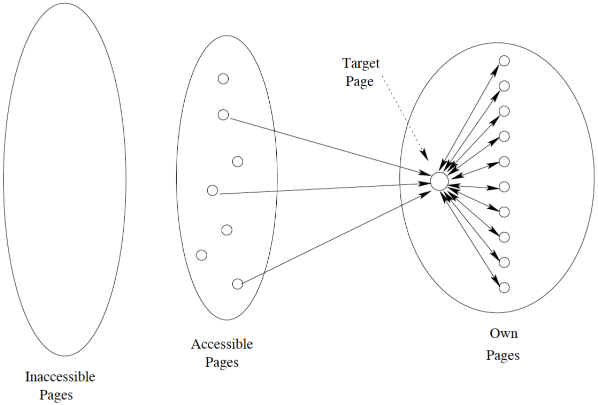
Types of pages

- *Inaccessible pages:* cannot be accessed by spammer; majority of pages
- *Accessible pages:* not owned, but can be accessed (manipulated)
  - ☞ Blogs, newspapers, forums allow leaving comments with links
- *Own pages:* owned and fully controlled by spammer

Spam farm

- Part of own pages with
  - *target page t*, for which maximum PageRank is to be achieved
  - *supporting pages m*, with links from and to t
- Note that without links from outside, spam farm would be useless

SPAMMER VIEW OF WEB



Spammer view: types of pages and spam farm

Adopted from `mmds.org`

Spam Farm: Analysis

- Let there be n web pages overall
- Let $\beta \in [0.8, 0.9]$ be the taxed fraction of PageRank
- Let there be a spam farm with target page t and m supporting pages
- Let $\text{In}(t)$ be all pages with a link to t; $\text{PR}(p)$ be the PageRank for a page p; $\text{Out}(p)$ be all successors of $p \in P$
- Let

$$x = \beta \sum_{p \in \text{In}(t)} \frac{\text{PR}(p)}{|\text{Out}(p)|}$$

  be the PageRank provided to t by accessible pages
- Let $y = \text{PR}(t)$ be the unknown PageRank of t
- The PageRank of each supporting page is

$$\beta \frac{y}{m} + \frac{(1-\beta)}{n}$$

  where $\beta \frac{y}{m}$ is due to t and $\frac{(1-\beta)}{n}$ is due to random teleporting

Spam Farm: Analysis

- Let $y = \mathrm{PR}(t)$ be the unknown PageRank of t
- Let x be the PageRank provided to t by accessible pages
- Let $\beta \frac{y}{m} + \frac{(1-\beta)}{n}$ be the PageRank of each supporting page

Solving for y

1. We compute

$$y = x + \beta m(\frac{\beta y}{m} + \frac{1-\beta}{n}) = x + \beta^2 y + \beta(1-\beta)\frac{m}{n} \tag{1}$$

2. This yields

$$y = \frac{x}{1-\beta^2} + c\frac{m}{n} \tag{2}$$

   where $c = \beta(1-\beta)/(1-\beta^2) = \beta/(1+\beta)$

Example: $\beta = 0.85$, so $1/(1-\beta^2) = 3.6$ and $c = 0.46$; spam farm has amplified external contribution to t by 360%; t also obtains 46% of the fraction m/n

Combating Link Spam

War on spam farms

- Search engines identify spam farm structures and eliminate pages from their index
- Spammers create alternative structures that raise PageRank of target pages
- Search engines in turn eliminate those structures, too
- ...
- Endless war between search engines and spammers

Systematic approaches

- *TrustRank:* Variation on topic-sensitive PageRank to lower score of spam pages
- *Spam mass:* Calculation that identifies pages likely to be spam
  ☞ Eliminate such pages or lower their PageRank substantially

TrustRank

- *TrustRank* is like topic-sensitive PageRank where the "topic" are pages believed to be "trustworthy"
  - Inaccessible pages belong to the topic
  - Accessible pages like blogs or newspapers are only borderline trustworthy
- Choosing trustworthy pages:
  1. Human picked pages, or pages of highest PageRank (not achievable by link spam)
  2. Pick pages trustworthy by domain, such as .edu, .ac.uk, .gov and so on

Spam Mass

Definition [Spam Mass]

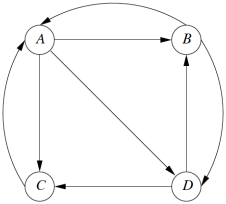
- For a page p, let $r(p)$ and $t(p)$ be its PageRank and its TrustRank
- The *spam mass* of p is defined to be

$$\frac{(r(p) - t(p))}{r(p)}$$

Explanation

- Negative or small spam mass indicates that p is not spam
- Spam mass close to 1 indicates that p is likely to be spam

Spam Mass: Example



Example web graph; B and D are trusted pages

Adopted from `mmds.org`

Node	PageRank	TrustRank	Spam Mass
A	$3/9$	$54/210$	0.229
B	$2/9$	$59/210$	-0.264
C	$2/9$	$38/210$	0.186
D	$2/9$	$59/210$	-0.264

Corresponding page rank, trust rank and spam mass

Adopted from `mmds.org`

Hubs and Authorities

Hubs and Authorities: Introduction

- The hubs-and-authorities algorithm, also called *HITS (hyperlink-induced topic search)*, is an alternative to PageRank
- *Similarities:*
  - Quantifies importance of pages
  - Involved fixedpoint computation by iterative matrix-vector multiplication
- *Differences:*
  - Divides pages into hubs and authorities
  - Not a preprocessing step: ranks importance of responses to query

HITS: Intuition

- Importance is twofold
- *Authorities* are pages deemed to be valuable because they provide information on a topic
  - E.g. course website at university
- *Hubs* are pages deemed to be valuable because of providing directions about topics
  - E.g. department directory providing links to all course websites
- Mutually recursive definition:
  - *Good hub* links to good authorities
  - *Good authority* is linked to by good hubs

Hubbiness and Authority: Definition

Definition [Hubbiness, Authority]

- Let the number of webpages be n
- Let $\mathbf{h} \in \mathbb{R}^n, \mathbf{a} \in \mathbb{R}^n$ be two vectors where
  - $\mathbf{h}_i$ quantifies the goodness of page i as a hub
  - $\mathbf{a}_i$ quantifies the goodness of page i as an authority
- $\mathbf{h}_i$ is also referred to as *hubbiness* of page i

Remark

- Values of $\mathbf{h}, \mathbf{a}$ are generally scaled such that
  - *either* the largest component is 1
  - *or* the sum of components is 1
  - In the following, first option will be used here

Link Matrix: Definition

Definition [Link Matrix]

- Let the number of webpages be n
- The *link matrix* $L \in \{0,1\}^{n \times n}$ of the Web is defined by

$$L_{ij} = \begin{cases} 1 & \text{there is a link from page } i \text{ to page } j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

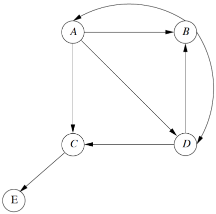
- Its transpose L^T is defined by $L^T_{ij} = L_{ji}$, that $L^T_{ij} = 1$ if there is a link from the j-th to the i-th page, and zero otherwise

Remark

- L^T is similar to the PageRank web matrix M insofar as

$$L^T_{ij} \neq 0 \quad \text{if and only if} \quad M_{ij} \neq 0$$

Link Matrix: Example



Example web graph

Adopted from `mmds.org`

$$L = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad L^{\mathrm{T}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Corresponding link matrix and its transpose

Adopted from `mmds.org`

Hubs and Authorities: Formal Relationship

- Good hub links to good authorities:

$$\mathbf{h}_i = \lambda \sum_{j=1}^{n} L_{ij}\mathbf{a}_j \qquad \text{or, equivalently} \qquad \mathbf{h} = \lambda L\mathbf{a} \tag{4}$$

  where λ represents the necessary scaling of $\mathbf{h}$

- Good authority is linked to by good hubs:

$$\mathbf{a}_i = \mu \sum_{j=1}^{n} L_{ij}^T\mathbf{h}_j \qquad \text{or, equivalently} \qquad \mathbf{a} = \mu L^T\mathbf{h} \tag{5}$$

  where μ represents the necessary scaling of $\mathbf{a}$.

Hubs and Authorities: Formal Relationship

- Substituting (5) into (4) yields:

$$\mathbf{h} = \lambda\mu LL^T\mathbf{h} \tag{6}$$

- Substituting (4) into (5) yields:

$$\mathbf{a} = \mu\lambda L^T L\mathbf{a} \tag{7}$$

- $\mathbf{h}$, $\mathbf{a}$ can be determined by solving linear equations
- *However:* $LL^T, L^T L$ are not sufficiently sparse for their size to allow for solving corresponding linear equations
- *Solution:* HITS algorithm

The HITS Algorithm

Initialization: Set $\mathbf{h}_i = 1$ for all i, that is $\mathbf{h} = (1, ..., 1)$

Iteration:

1. Compute
$$\mathbf{a} = L^T\mathbf{h}$$
2. Scale such that largest component of $\mathbf{a}$ is 1
3. Compute
$$\mathbf{h} = L\mathbf{a}$$
4. Scale such that largest component of $\mathbf{h}$ is 1
5. Repeat until convergence

HITS ALGORITHM: EXAMPLE

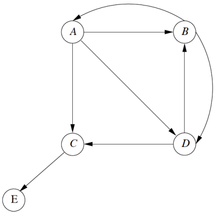
$$
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}
\quad
\underset{L^{\mathrm{T}}\mathbf{h}}{\begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}}
\quad
\underset{\mathbf{a}}{\begin{bmatrix} 1/2 \\ 1 \\ 1 \\ 1 \\ 1/2 \end{bmatrix}}
\quad
\underset{L\mathbf{a}}{\begin{bmatrix} 3 \\ 3/2 \\ 1/2 \\ 2 \\ 0 \end{bmatrix}}
\quad
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 1/2 \\ 1/6 \\ 2/3 \\ 0 \end{bmatrix}}
$$

$$
\underset{L^{\mathrm{T}}\mathbf{h}}{\begin{bmatrix} 1/2 \\ 5/3 \\ 5/3 \\ 3/2 \\ 1/6 \end{bmatrix}}
\quad
\underset{\mathbf{a}}{\begin{bmatrix} 3/10 \\ 1 \\ 1 \\ 9/10 \\ 1/10 \end{bmatrix}}
\quad
\underset{L\mathbf{a}}{\begin{bmatrix} 29/10 \\ 6/5 \\ 1/10 \\ 2 \\ 0 \end{bmatrix}}
\quad
\underset{\mathbf{h}}{\begin{bmatrix} 1 \\ 12/29 \\ 1/29 \\ 20/29 \\ 0 \end{bmatrix}}
$$

First two iterations of HITS algorithm

Adopted from `mmds.org`

HITS Algorithm: Example



A and D are good hubs, B and C are good authorities

Adopted from `mmds.org`

$$\mathbf{h} = \begin{bmatrix} 1 \\ 0.3583 \\ 0 \\ 0.7165 \\ 0 \end{bmatrix} \qquad \mathbf{a} = \begin{bmatrix} 0.2087 \\ 1 \\ 1 \\ 0.7913 \\ 0 \end{bmatrix}$$

Limits of **h**, **a** on graph

Adopted from `mmds.org`

Frequent Itemsets
Introduction

Frequent Itemsets: Overview

Foundations

- There are *items* available in the market
- There are *baskets*, sets of items having been purchased together
- A *frequent itemset* is a set of items that is found to commonly appear in many baskets
- The *frequent-itemset problem* is to identify frequent itemsets

Market-Basket Model

Market-basket model

- The market-basket model is a *many-many-relationship*
  - One basket holds many items
  - One item appears in several baskets
- Each basket is an itemset, i.e. a set of (one or several) items
- Usually, the number of items in a basket is small compared to number of items overall
- Number of baskets is usually large; too large to fit in main memory
- Data usually is a sequence of baskets

Frequent Itemsets: Definition

Definition [Frequent Itemset]:

- Let $s > 0$ be a *support threshold*
- Let I be a set of items
- $\text{supp}(I)$, the *support* of I, is the number of baskets in which I appears as a subset

An itemset I is referred to as *frequent* if

$$\text{supp}(I) \geq s \tag{8}$$

that is, if the support of I is at least the support threshold

Frequent Itemsets: Example

Baskets

1. {and, dog, bites}
2. {news, claims, a, cat, mated, with, a, dog, and, produced, viable, offspring}
3. {cat, killer, likely, is, a, big, dog}
4. {professional, free, advice, on, dog, training, puppy, training}
5. {cat, and, kitten, training, behavior}
6. {dog, cat, provides, training, in, Oregon}
7. {dog, and, cat, is, a, slang, term, used, by, police, officers, for, a, male-female, relationship}
8. {shop, for, your, show, dog, grooming, and, pet, supplies}

- E.g. supp({dog}) = 7, supp({and}) = 5, supp({dog, and}) = 4
- Let the support threshold $s = 3$
- 5 frequent singletons: {dog},{cat},{a},{and},{training}
- 5 frequent doubletons: {dog, a},{dog, and},{dog, cat},{cat, a},{cat, and}
- 1 frequent triple: {dog, cat, a}

Frequent Itemsets: Applications

- *Retailers / Supermarkets / Chain stores*
  - *Items:* Products offered
  - *Baskets:* Sets of products purchased by one customer during one shopping run
  - *Frequent Itemsets:* Products purchased together unusually often
    ☞ Beer and diapers
- *Related concepts*
  - *Items:* Words, excluding stop words
  - *Baskets:* News articles, documents
  - *Frequent Itemsets:* Groups of words representing joint concept
- *Plagiarism*
  - *Items:* Documents
  - *Baskets:* Sentences
  - *Frequent Itemsets:* Documents containing unusually many sentences in common

Association Rules

- Let j be an item and I be an itemset
- An association rule
$$I \to j$$
expresses that if I is likely to appear in a basket, so is j
- In other words, if I shows in basket, one is confident to assume that j does, too

Definition [Confidence]:
The *confidence* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} \tag{9}$$

that is the fraction of I containing baskets that also contain j.

Association Rules: Confidence

Definition [Confidence]:
The *confidence* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)}$$

that is the fraction of I containing baskets that also contain j.

Example from above

- Confidence of $\{cat, dog\} \to and$ is $3/5$
- Confidence of $\{cat\} \to kitten$ is $1/6$

Association Rules: Interest

- Let n be the number of baskets overall
- Confidence for $I \to j$ can be meaningless if fraction of baskets containing j is large
- Confidence may just reflect that fraction
- So presence of I does not increase confidence to see j as well
- *Interest* is supposed to put this into context

Definition [Interest]:
The *interest* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} - \frac{\text{supp}(\{j\})}{n} \tag{10}$$

that is the confidence of $I \to j$ minus the fraction of baskets that contain j

Association Rules: Interest

Definition [Interest]:
The *interest* of a rule $I \rightarrow j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} - \frac{\text{supp}(\{j\})}{n}$$

that is the confidence of $I \rightarrow j$ minus the fraction of baskets that contain j

Examples

- $\{diapers\} \rightarrow beer$ was found to have great interest
- $\{dog\} \rightarrow cat$ has interest $5/7 - 3/4 = -0.036$
- $\{cat\} \rightarrow kitten$ has interest $1/6 - 1/8 = 0.042$

Frequent Itemsets to Association Rules

Situation

- Consider frequent itemsets of "reasonably high" support s
  - Note that each frequent itemset suggests to be acted upon
    ☞ keep their number reasonably low
  - Reasonably high often means about 1% of baskets
- Confidence for a rule $I \to j$ should be at least (about) 50%
  ☞ Support for $I \cup \{j\}$ also fairly high

Procedure

- Assume all I with $\text{supp}(I) \geq s$ have been mined
- For J of n items with $\text{supp}(J) \geq s$, there are n possible association rules $J \setminus \{j\} \to J$
- $\text{supp}(J) \geq s$ implies $\text{supp}(J \setminus \{j\}) \geq s$
- Confidence of $J \setminus \{j\} \to J$ is easily computed as

$$\frac{\text{supp}(J)}{\text{supp}(J \setminus \{j\})}$$

Mining Frequent Itemsets
The A-Priori Algorithm

Market-Basket Data: Representation

- Market-basket data is stored in a file basket-by-basket
  - If items refer to identifiers, for example $\{3, 36, 99\}\{6, 78, 11\}\ldots$
- *Assumption:* Average size of basket is rather small
- *Usually,* file does not fit in main memory
- Generating all subsets of size k for a basket of size n requires

$$\binom{n}{k} \approx \frac{n^k}{k!}$$

  runtime
- This often is little time because
  - n was assumed to be small
  - k is usually very small
  - When k is large, one can virtually reduce n further by removing infrequent items

Market-Basket Data: Runtime Consideration

Insight

- Runtime is dominated by transferring data from disk to main memory
- *Consequence:* Processing all baskets is proportional to size of file
- *Runtime of algorithm* is proportional to number of passes through file
- For a *fast frequent itemset mining* algorithm:

Limit number of passes through basket file

Use of Main Memory

- *Issue:* One needs to store counts for itemsets of size k
  - There could be many such itemsets
  - How to store these counts?
- *Consequence:* There is a limit on the number of items an algorithm can deal with
- *Example:*
  - Let there be n items
  - For counting pairs, we need to store $\binom{n}{2} \approx n^2/2$ counts
  - Integers of 4 bytes: need $2n^2$ bytes to store counts
  - Consider machine of 2 GB, or $\approx 2^{31}$ bytes of main memory
  - Then $n < 2^{15} \approx 33\,000$ is required
- *Note:* Items can be hashed to integers, if they are not integers

Storing Itemset Counts: The Triangular-Matrix Method

- In the following, consider storing itemsets of size 2
  - Remember that support threshold is quite large in real applications
  - So, many more pairs than triples, quadruples and so on in real applications
- *Insight:* Storing counts $a[i,j]$ in matrix $A = (a[i,j])_{1 \leq i < j \leq n} \in \mathbb{N}^{n \times n}$ wastes half of A
- *Solution:* Store count for pair of items $\{i,j\}, 1 \leq i < j \leq n$ in

$$a[k] \quad \text{where} \quad k = (i-1)(n - \frac{i}{2}) + j - i \tag{11}$$

This stores pairs in lexicographical order

$$\{1,2\}, \{1,3\}, ..., \{1,n\}, \{2,3\}, ..., \{2,n\}, ..., \{n-2,n\}, \{n-1,n\}$$

Storing Itemset Counts: The Triples Method

- Store triples $[i, j, c]$ for all pairs $\{i, j\}$ whose count $c > 0$
- For example, do this with hash table, hashing i, j as search key
- *Advantage:* Does not require space for pairs $\{i, j\}$ of count zero
- *Disadavantage:* Requires three times the space if $c > 0$
- *Rationale:* Triangular matrix method better if at least 1/3 of the $\binom{n}{2}$ pairs appear in basket

Storing Itemset Counts: Example

Example

- Consider
  - 100 000 items
  - 10 000 000 baskets of
  - 10 items each
- Triangular-matrix method: $\binom{10^5}{2} \approx 5 \times 10^9$ integer counts
- Triples method: $10^7 \binom{10}{2} \approx 4.5 \times 10^8$ counts, making for $3 \times 4.5 \times 10^8 = 1.35 \times 10^9$ integers to be stored
- Triples method proves to be more appropriate

MONOTONICITY

THEOREM [MONOTONICITY]:

- Let s be the support threshold.
- Let I, J be sets such that $J \subseteq I$

Then if I is frequent, any subset J of I is, too:

$$\text{supp}(I) \geq s \quad \text{implies} \quad \text{supp}(J) \geq s \tag{12}$$

PROOF.
Each basket that holds I also holds J, as J is contained in I. So, the number of baskets that hold J is at least as large as the number of baskets that hold I. □

Maximal Frequent Itemset

Definition [Maximal Frequent Itemset]:

- Let s be the support threshold.
- Let I be frequent, that is $\text{supp}(I) \geq s$.

I is said to be *maximal* if no superset of I is frequent:

$$\text{for all } J \supsetneq I : \text{supp}(J) < s \tag{13}$$

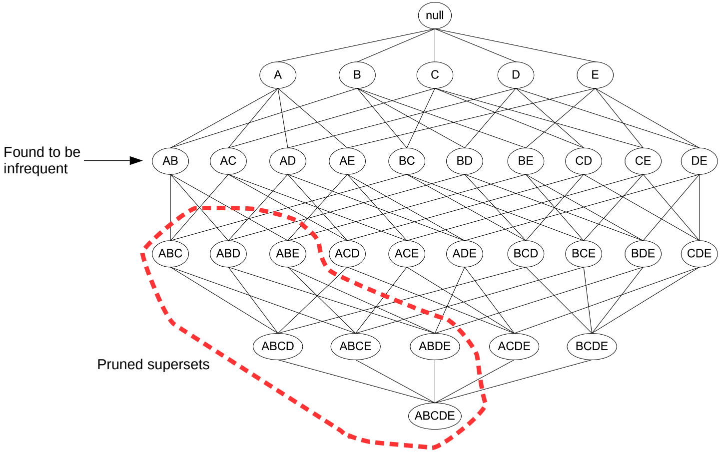
Example (from above):

- At support threshold $s = 3$, we found frequent pairs $\{dog, a\}, \{dog, and\}, \{dog, cat\}, \{cat, a\}, \{cat, and\}$
- $\{dog, cat, a\}$ was found the only frequent triple
- ☞ $\{dog, cat, a\}, \{dog, and\}$ and $\{cat, and\}$ are maximal, while $\{dog, a\}, \{dog, cat\}, \{cat, a\}$ are not

Note on Counting Pairs

- The number of frequent pairs is larger than frequent triples, quadruples, ... why?
- For making sense, number of (maximal) frequent itemsets is supposed to be sufficiently small
  - Human applicants need to work it out on all of them
- So, support threshold is set sufficiently high
- Any maximal frequent itemset holds many more smaller, non-maximal frequent itemsets
- The resulting situation implies that there are many more frequent pairs than triples, many more frequent triples than quadruples, and so on
- *Important:*
  - Still, the possible number of triples, quadruples is (much) greater than pairs
  - Any good frequent itemset *algorithm needs to avoid running through all possible triples, quadruples, and so on*

Monotonicity to the Rescue



Itemsets for items A,B,C,D,E

Neglecting supersets of infrequent pair {A,B}

Adopted from `mmds.org`

A-Priori Algorithm: Motivation

In the following, we focus on determining frequent pairs.

*N*aive Approach

Consider the algorithm

- For each basket, use double loop to generate all pairs contained in it
- For each pair generated, add 1 to its count
- Store counts using triangular or triples method
- At the end, run through all pairs and determine those whose counts exceed support threshold s
- *Benefit:* Only one pass through all baskets
- *Issue:* Number of pairs considered usually does not fit in main memory

A-Priori Algorithm: Motivation

In the following, we focus on determining frequent pairs.

*N*aive Approach

- *Possible Benefit:* Only pass through all baskets
- *Issue:* Number of pairs considered usually does not fit in main memory

*S*olution: A-Priori-Algorithm

- Have *two passes through baskets* instead of one
- In first run, determine candidate pairs, for which counts are stored
- In second run, determine counts for candidate pairs
- Finally filter for frequent pairs

A-PRIORI ALGORITHM: FIRST PASS

Create and Maintain Two Tables

- *First table A:* Let x be an item name, then $A[x]$ reflects that x is the $A[x]$-th item in the order of their appearance in the basket file
- *Second table B:* Let k be an item number, then $B[k]$ is the number of baskets in which item number k appears

Read Baskets: Fill Table B

- For each basket, for each item x in the basket, do

$$B[A[x]] = B[A[x]] + 1 \tag{14}$$

- That is, iteratively increase item counts while running through all items in all baskets

A-PRIORI ALGORITHM: SECOND PASS I

- Let n be the number of items
- Let m be the number of items found to be *frequent*
- By user constraints, usually $m << n$

Create Third Table

- *Third table C:* Let $1 \leq k \leq n$ be an item number. Then

$$C[k] = \begin{cases} 0 & \text{if item number } k \text{ is not frequent} \\ l & \text{if item number } k \text{ was found the } l\text{-th frequent item} \end{cases} \tag{15}$$

So, $C \in \{0, 1, ..., m\}^n$, where

- $C[k] = 0$ $n - m$ times
- $C[k] = i, 1 \leq i \leq m$ exactly one time
- $0 < C[k_1] < C[k_2]$ implies $k_1 < k_2$, expressing that C preserves the order of appearance of items

A-Priori Algorithm: Second Pass II

Count Pairs Data Structure

- Use either triangular or triples method data structure to hold counts
  - For using triangular method, renumbering necessary
- By monotonicity, a pair can only be frequent, if both items are frequent
- So, space required is $O(m^2)$ rather than $O(n^2)$
  ☞ $m << n$ implies $m^2 << n^2$, so fits in main memory!

Examine Baskets

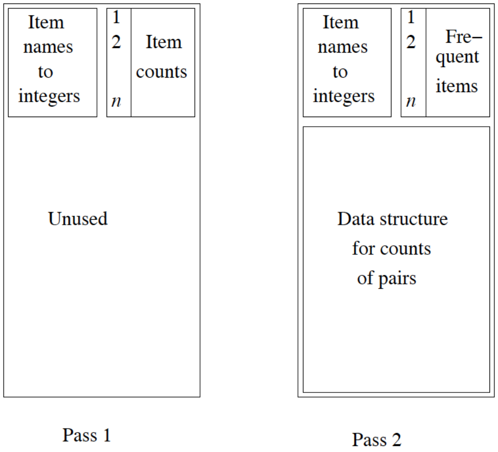
1. For each basket, for each item x, see whether

$$C[A[x]] > 0 \quad \text{that is, whether } x \text{ is frequent} \tag{16}$$

2. Using double loop, generate all pairs of frequent items in the basket
3. For each such pair, increase count by one in pair count data structure

Eventually: examine which pairs are frequent in pair count data structure

A-Priori Algorithm: Main Memory Usage



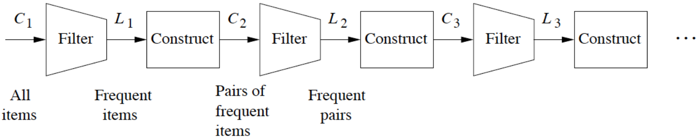
Use of main memory during A-Priori passes

Adopted from `mmds.org`

A-Priori Algorithm: All Frequent Itemsets

- *One extra pass* for each $k > 2$ to mine frequent itemsets of size k
- The A-Priori algorithm proceeds iteratively
  - Mining frequent itemsets of size $k + 1$ is based on knowing frequent itemsets of size k
- Each iteration consists of two steps for each k:
  - Generate a candidate set C_k
  - Filter candidate set C_k to produce L_k, the truly frequent itemsets of size k
- The algorithm terminates at first k where L_k is empty
  - Monotonicity says we are done mining frequent itemsets

A-Priori Algorithm: Candidate Generation and Filtering



A-Priori algorithm: Alternating between candidate generation and filtering

Adopted from `mmds.org`

- *Construct:* Let C_k be all itemsets of size k, every $k-1$ of which belong to L_{k-1}
- *Filter:* Make a *pass through baskets* to count members of C_k; those with count exceeding s will be part of L_k
  - For storing counts for itemsets of size k, extend triples method
  - E.g. storing quadruples for frequent triples, and so on...

MATERIALS / OUTLOOK

- See *Mining of Massive Datasets*, sections 5.4, 5.5, 6.1, 6.2
- As usual, see `http://www.mmds.org/` in general for further resources
- Next lecture: 'Frequent Itemsets II / Recommendation Systems"
  - See *Mining of Massive Datasets*, 6.3, 6.4.5, 9.1, 9.2