

# Association Rules, Min-Hashing and Nearest Neighbors for Sparse Vectors

Yury Person

IPAM & Humboldt University Berlin

October 31, 2007

# Outline

Association Rules

Min-Hashing

Nearest Neighbors for Sparse Vectors

# Problem Introduction

A database of some retail company consists of transactions which contain items bought together. The question is to derive frequently bought itemsets and relations among them.

## Example

Many people buy bread, butter and milk together. An association rule would be  $\{bread, butter\} \rightarrow milk$ , if most the transactions, when bread and butter was bought, also contained item *milk*.

# Formal Statement of the problem I

- ▶ A set  $\mathcal{I}$  of  $m$  items :  $\{i_1, \dots, i_m\}$ .
- ▶ A family  $\mathcal{D}$  of transactions:  $\forall T \in \mathcal{D} T \subseteq \mathcal{I}$ .
- ▶  $s, c \in [0, 100]$

Here we consider only simple association rules:

**Definition (Association Rule)**

$X \rightarrow \{y\}$ , where  $X \subseteq \mathcal{I}$  and  $y \in \mathcal{I}$ .

# Formal Statement of the problem II

## Definition (Confidence& Support)

- ▶ We say that an association rule  $X \rightarrow \{y\}$  has confidence at least  $c$ , if  $c\%$  of transactions in  $\mathcal{D}$  that contain  $X$ , also contain  $y$ .
- ▶ The rule  $X \rightarrow \{y\}$  has support  $s$  in the transaction set  $\mathcal{D}$  if at least  $s\%$  of all transactions contain  $X \cup \{y\}$ .

## Problem

*Find all association rules in  $\mathcal{D}$  with support at least  $s$  and confidence  $c$ .*

## Problem Decomposition: 2 steps

1. Find all large itemsets, i.e. those of support at least  $s$ .
2. Generate from these large itemsets all association rules that have confidence at least  $c$ .

## Problem Decomposition: 2 steps

1. Find all large itemsets, i.e. those of support at least  $s$ .
2. Generate from these large itemsets all association rules that have confidence at least  $c$ .

The second step is straightforward!

## First Step: Algorithm *Apriori*

```
1 initialization:  $L_1 = \{\text{large 1-itemsets}\}$ ;  
2 for ( $k = 2$ ;  $L_{k-1} \neq \emptyset$ ;  $k++$ ) do  
3    $C_k = \text{apriori-gen}(L_{k-1})$ ; //New candidates;  
4   for all transactions  $T \in \mathcal{D}$  do  
5      $C_T = \text{subset}(C_k, T)$ ; //Candidates contained in  $T$ ;  
6     for all candidates  $c \in C_T$  do  
7        $c.\text{count}++$ ;  
8     end  
9   end  
10   $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$ ;  
11 end  
12 Answer =  $\cup_k L_k$ ;
```



## Candidate Generation apriori-gen

► Join step

insert into  $C_k$ ;

select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$ ;

from  $L_{k-1}p, L_{k-1}q$ ;

where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} <$   
 $q.item_{k-1}$ ;

## Candidate Generation apriori-gen

- ▶ Join step  
insert into  $C_k$ ;  
select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$ ;  
from  $L_{k-1}p, L_{k-1}q$ ;  
where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$ ;
- ▶ Prune step  
**for every itemsets  $c \in C_k$  do**  
    **for every  $(k-1)$ -subset  $s$  of  $c$  do**  
        **if  $(s \notin L_{k-1})$  then**  
            delete  $c$  from  $C_k$   
        **end**  
    **end**  
**end**

The procedure generates a superset of the set of all large  $k$ -itemsets.

## Example

- ▶ Let  $L_3 = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{1, 3, 5\}, \{2, 3, 4\}\}$
- ▶ After the Join step:  $C_4 = \{\{1, 2, 3, 4\}, \{1, 3, 4, 5\}\}$
- ▶ The Prune step deletes the itemset  $\{1, 3, 4, 5\}$  because  $\{1, 4, 5\} \notin C_4$

# Modifications of the Algorithm

- ▶ generalize association rules to  $X \rightarrow Y$  with  $X, Y \subset \mathcal{I}, X \cap Y = \emptyset$
- ▶ speed-up by testing only transactions  $T \in \mathcal{D}$  that make sense
- ▶ No running time guarantees, but good performance in practice

# Identifying Duplicates on the Web & Document similarity

## Problem

*Given a copy of the web. Identify near duplicates of the web pages.*

# Identifying Duplicates on the Web & Document similarity

## Problem

*Given a copy of the web. Identify near duplicates of the web pages.*

## Idea

*First compute sketches of every document. Every sketch is small.  
Introducing appropriate measure (Jaccard) identify duplicates.*

# $\omega$ -shingling

## Definition

Given a document  $\mathcal{D}$ . A contiguous subsequence of  $\omega$  words in  $\mathcal{D}$  is called an  $\omega$ -shingle.  $\omega$ -shingling of  $\mathcal{D}$  is a (multi-)set  $S(\mathcal{D}, \omega)$  of all  $\omega$ -shingles in  $\mathcal{D}$

# $\omega$ -shingling

## Definition

Given a document  $\mathcal{D}$ . A contiguous subsequence of  $\omega$  words in  $\mathcal{D}$  is called an  $\omega$ -shingle.  $\omega$ -shingling of  $\mathcal{D}$  is a (multi-)set  $S(\mathcal{D}, \omega)$  of all  $\omega$ -shingles in  $\mathcal{D}$

## Example

▶  $\mathcal{D} = (a, \text{rose}, \text{is}, a, \text{rose}, \text{is}, a, \text{rose})$

▶ 4-shingling is

$$S(\mathcal{D}, 4) = \{(a, \text{rose}, \text{is}, a), (\text{rose}, \text{is}, a, \text{rose}), (\text{is}, a, \text{rose}, \text{is})\}$$



# Resemblance, Containment

Given two documents,  $A$  and  $B$ . We fix a shingle of size  $\omega$ .

## Definition

We call

$$r_{\omega}(A, B) = \frac{|S(A, \omega) \cap S(B, \omega)|}{|S(A, \omega) \cup S(B, \omega)|}$$

the resemblance of  $A$  and  $B$ .

# Resemblance, Containment

Given two documents,  $A$  and  $B$ . We fix a shingle of size  $\omega$ .

## Definition

We call

$$r_\omega(A, B) = \frac{|S(A, \omega) \cap S(B, \omega)|}{|S(A, \omega) \cup S(B, \omega)|}$$

the resemblance of  $A$  and  $B$ .

and

## Definition

We call

$$c_\omega(A, B) = \frac{|S(A, \omega) \cap S(B, \omega)|}{|S(A, \omega)|}$$

the containment of  $A$  in  $B$ .

The resemblance captures the notion of 'roughly the same'!

# Computing Sketches

Let  $\Omega$  be a universe of all shingles of size  $\omega$ . Assume that  $\Omega$  is totally ordered. Further let  $MIN_s(A)$  denote the  $s$  smallest elements of  $A$ .

## Definition

Given the document  $A$ ,  $s \in \mathbb{N}$  and  $\pi : \Omega \rightarrow \Omega$  permutation chosen uniformly at random. We define the sketch  $M(A)$  of  $A$  of size  $s$  to be the  $s$  smallest elements among  $A$  under  $\pi$ :

$$M(A) = MIN_s\{\pi(A)\}$$

# Computing Sketches

Let  $\Omega$  be a universe of all shingles of size  $\omega$ . Assume that  $\Omega$  is totally ordered. Further let  $MIN_s(A)$  denote the  $s$  smallest elements of  $A$ .

## Definition

Given the document  $A$ ,  $s \in \mathbb{N}$  and  $\pi : \Omega \rightarrow \Omega$  permutation chosen uniformly at random. We define the sketch  $M(A)$  of  $A$  of size  $s$  to be the  $s$  smallest elements among  $A$  under  $\pi$ :

$$M(A) = MIN_s\{\pi(A)\}$$

## Theorem (A.Broder'1997)

$$r_\omega(A, B) = \frac{|MIN_s(M(A) \cup M(B)) \cap M(A) \cap M(B)|}{|MIN_s(M(A) \cup M(B))|}$$

# Min-wise Independent families

## Problem

*In practice, it is impossible to choose  $\pi$  uniformly at random!*

# Min-wise Independent families

## Problem

*In practice, it is impossible to choose  $\pi$  uniformly at random!*

## Definition (Min-wise independent family)

A family  $\mathcal{F} \subseteq S_n$  is min-wise independent if for any set  $X \subset \{1, \dots, n\}$  and any  $x \in X$ , when  $\pi$  is chosen at random in  $\mathcal{F}$  we have

$$\Pr(\min(\pi(X)) = \pi(x)) = \frac{1}{|X|}$$

# Min-wise Independent families

## Problem

*In practice, it is impossible to choose  $\pi$  uniformly at random!*

## Definition (Min-wise independent family)

A family  $\mathcal{F} \subseteq S_n$  is min-wise independent if for any set  $X \subset \{1, \dots, n\}$  and any  $x \in X$ , when  $\pi$  is chosen at random in  $\mathcal{F}$  we have

$$\Pr(\min(\pi(X)) = \pi(x)) = \frac{1}{|X|}$$

## Definition ( $\varepsilon$ -approximately min-wise independent family)

A family  $\mathcal{F} \subseteq S_n$  is  $\varepsilon$ -approximately min-wise independent if for any set  $X \subset \{1, \dots, n\}$  and any  $x \in X$ , when  $\pi$  is chosen at random in  $\mathcal{F}$  we have

$$\left| \Pr(\min(\pi(X)) = \pi(x)) - \frac{1}{|X|} \right| \leq \frac{\varepsilon}{|X|}$$

# Bounds for Minimum Size Families I

## Theorem

$\mathcal{F}$  is at least as large as the least common multiple of the numbers  $1, 2, \dots, n$  and hence  $|\mathcal{F}| \geq e^{n-o(n)}$ .

## Proof.

- ▶ take any subset  $X$  of  $\{1, \dots, n\}$ ,  $|X| = j$
- ▶ every element of  $X$  must be the minimum under  $\mathcal{F}$  the same number of times, so  $j$  divides  $|\mathcal{F}|$
- ▶ use *Prime Number Theorem* to derive the lower bound of  $e^{n-o(n)}$





# Bounds for Minimum Size Families II

## Theorem

*There exists  $\mathcal{F}$  of size*

$$\prod_{i=1}^{\lceil \log n \rceil} \binom{\lceil n/2^{i-1} \rceil}{\lceil n/2^i \rceil}$$

# Bounds for Minimum Size Families II

## Theorem

*There exists  $\mathcal{F}$  of size*

$$\prod_{i=1}^{\lceil \log n \rceil} \binom{\lceil n/2^{i-1} \rceil}{\lceil n/2^i \rceil}$$

## Exercise

*Prove that this bound is divisible by the least common multiple of the first  $n$  natural numbers.*

# Existential Upper Bound for $\varepsilon$ -approximate Families

## Theorem

*There exist families of size  $O(\frac{n^2}{\varepsilon^2})$  that are approximately minwise independent with high probability.*

In practice, one cannot conveniently represent a random permutation!

# Existential Upper Bound for $\varepsilon$ -approximate Families

## Theorem

*There exist families of size  $O(\frac{n^2}{\varepsilon^2})$  that are approximately minwise independent with high probability.*

In practice, one cannot conveniently represent a random permutation!

## Problem

*Construct such family!*

# Existential Upper Bound for $\varepsilon$ -approximate Families

## Theorem

*There exist families of size  $O(\frac{n^2}{\varepsilon^2})$  that are approximately minwise independent with high probability.*

In practice, one cannot conveniently represent a random permutation!

## Problem

*Construct such family!*

One tries families of linear permutations which behave good in practice.

# Problem Introduction

## Problem

*There are  $n$  people and  $m$  books. Every person likes exactly  $k$  books.*

*Given another person  $Q$  that likes  $k$  books, find a person in the database that likes maximum possible number of books.*

### **Constraints:**

- ▶  $k \ll n, m$
- ▶ query time is  $\text{poly}(k, \log n)$
- ▶ preprocessing time:  $\text{poly}(k, n, m)$

# Our Approach I

Utilize the idea of characteristic itemsets:

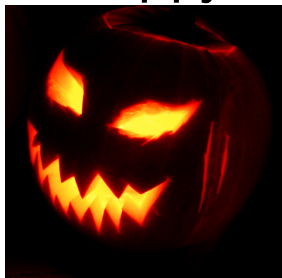
- ▶ there are  $O(\text{poly}(k) * n)$  characteristic itemsets (of books)
- ▶ every person likes at least one characteristic itemset
- ▶ every characteristic itemset is appreciated by  $\text{poly}(k)$  persons
- ▶ every person shares at least one characteristic itemset with each of its nearest neighbors

## Our Approach II

1. Given database, extract  $O(n)$  characteristic itemsets
2. from the query  $Q$  distill characteristic itemsets
3. compute nearest neighbors for  $Q$



Thank you & Happy



Halloween!