ECE20008-01 Project Practice 1

Finding Frequent-Item Sets

26 Sep, 2017

Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University
http://www.mmds.org

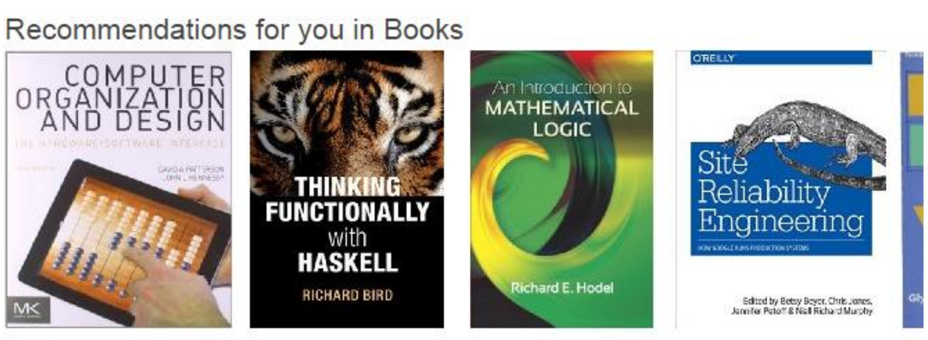
Taken partly from *Mining of Massive Datasets* and then edited by Shin Hong

Data Mining*

- Finding rules/models that summarize the given data
  - Discovering hidden knowledge about the target domain
- Use statistical analyses on large data set
  - Simple data model
  - Scalability is the major concern

Finding Frequent-Itemsets

- What sets of items are often bough together in marketplaces?
  - People who buy one item are likely to buy another related items
  - E.g. bread and milk, chicken and coke, beer and diaper
  - E.g. associated keyword
  - E.g. 35% of product sales of Amazon.com result from its recommendation system (2006)



amazon.com

Chapter 6 of the "Mining Massive Datasets" Course @ Standford http://www.mmds.org/

Market-Basket Model

- A market has *items*
- A *basket* contains a subset of items (i.e., *itemset*)
  - Usually, the number of items in a basket is small
- A set of items is *frequent* if there are *s* or more baskets that contain the set of items
  - *s*: support threshold

Example: Movies

Bourne Identity

Bourne Supremacy

Bourne Ultimatum

Henry Potter I

Henry Potter II

Henry Potter III

	BI	BS	BU	$HP1$	$HP2$	$HP3$
V_1	x	x	x			
V_2				x	x	x
V_3	x			x		
V_4	x	x		x	x	
V_5	x	x	x	x		
V_6	x		x			
V_7		x		x	x	
V_8	x	x		x	x	x

s = 3

n=1: {BI}, {BS}, {BU}, {HP1}, {HP2}

n=2: {BI, BS}, {BI, BU}, {BI, HP1}, {BS, HP1}, {BS, HP2}, {HP1, HP2}

n=3: {BI, BS, HP1}

Example: Words

- A basket contains words appearing in one document

1. {Cat, and, dog, bites}
2. {Yahoo, 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, and, behavior}
6. {Dog, &, Cat, provides, dog, training, in, Eugene, 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}

	training	a	and	cat
dog	4, 6	2, 3, 7	1, 2, 8	1, 2, 3, 6, 7
cat	5, 6	2, 3, 7	1, 2, 5	
and	5	2, 7		
a	none			

Association Rule

- For a set of items I and an item j,

$$I \rightarrow j$$

  holds when all of the items in I appear in some baskets, then j is likely to appear in the same basket as well.

- The *confidence* of rule $I \rightarrow j$ is the ratio of the supports for $I \cup \{j\}$ (i.e., the number of baskets with $I \cup \{j\}$) to the supports for I.

- The *interest* of rule $I \rightarrow j$ is the confidence of the rule minus the faction of the all baskets that contains *j*

Example: Movie

	BI	BS	BU	$HP1$	$HP2$	$HP3$
V_1	x	x	x			
V_2				x	x	x
V_3	x			x		
V_4	x	x		x	x	
V_5	x	x	x	x		
V_6	x		x			
V_7		x		x	x	
V_8	x	x		x	x	x

- **{BI, BS} → BU**
  - Confidence: $|\{V_1, V_5\}|$ / $|\{V_1, V_4, V_5, V_8\}| = 0.5$
  - Interest: $0.5 - |\{V_1, V_4, V_6\}| / 8 = 0.125$

Example: Movie

	BI	BS	BU	$HP1$	$HP2$	$HP3$
V_1	x	x	x			
V_2				x	x	x
V_3	x			x		
V_4	x	x		x	x	
V_5	x	x	x	x		
V_6	x		x			
V_7		x		x	x	
V_8	x	x		x	x	x

- **{BI, BS} → BU**
  - Confidence: $|\{V_1, V_5\}|$ / $|\{V_1, V_4, V_5, V_8\}| = 0.5$
  - Interest: $0.5 - |\{V_1, V_5, V_6\}| / 8 = 0.125$
- **{HP1, HP2} → HP3**
  - Confidence: $|\{V_2, V_8\}| / |\{V_2, V_4, V_7, V_8\}| = 0.5$
  - Interest: $0.5 - |\{V_2, V_8\}| / 8 = 0.25$

Counting Itemsets

- It is desirable to maintain the itemsets and their counts in main memory throughout analysis
  - E.g. the memory required for counting doubletons (pairs) of *n* items
    - The number of pairs: C(n, 2) = n (n – 1) / 2
    - If each count takes 4 bytes, the counting takes roughly $2n^2$ bytes
    - 2 GB can holds n $\leq 2^{15}$
- The Triangular-Matrix method
  - `a[i,j] = a + (i – 1)(n – i /2) + j - i`
- The Triples method
  - Map a count to the pair of *i* and *j*.
  - Use hash functions to reduce space complexity.

The A-Priori Algorithm (1/4)

- Monotonicity
  - If an itemset *I* is frequent (i.e., there exists more than s supports), then so is every subset of *I*
  - If an itemset *K* appears less than *s* times, then none of its superset appears more than *s* times
- Idea
  - Find the frequent itemsets from ones with size 1 and then increase the size by one at a time
  - Count for an itemset of size *n* whose all subsets of size *n*-1 are frequent itemsets
    - If an itemset *K* is not frequent at size *i*, no need to count for any superset of *K* at size *i*+1

The A-Priori Algorithm (2/4)

- The first pass: finding frequent singleton itemsets
  1. Translate each item name into a unique integer
  2. For each basket,

     for each item, increase the count of the corresponding singleton itemset by 1.
  3. Transfer to the second pass only items whose counts are larger than or equal to s.

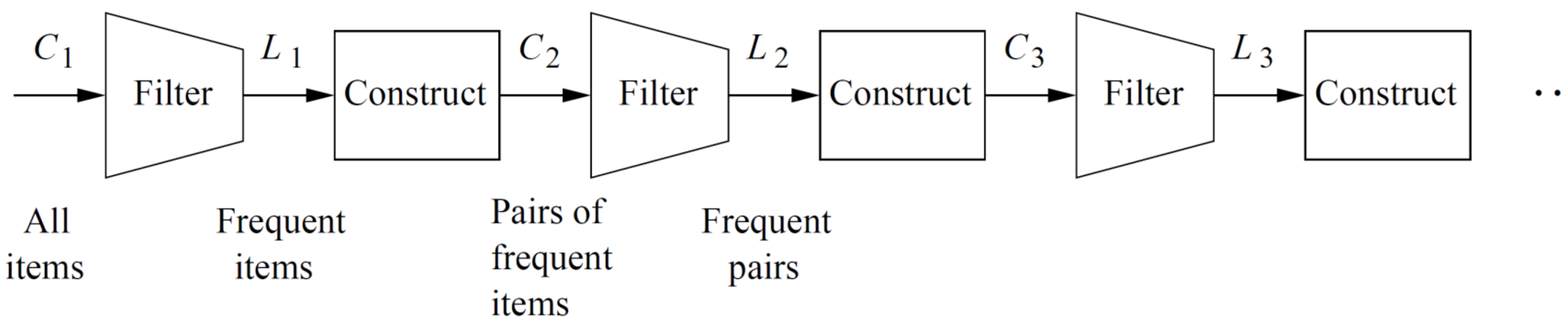
The A-Priori Algorithm (3/4)

- The second pass: finding frequent doubletons (pairs)
  1. Assign new identifiers to the selected singletons
  2. Generate all pairs of the selected singletons
  3. For each basket,
     for each pair of items,
     increase the count of the doubleton itemset by 1
     if the corresponding doubleton itemset was generated.

The A-Priori Algorithm (4/4)

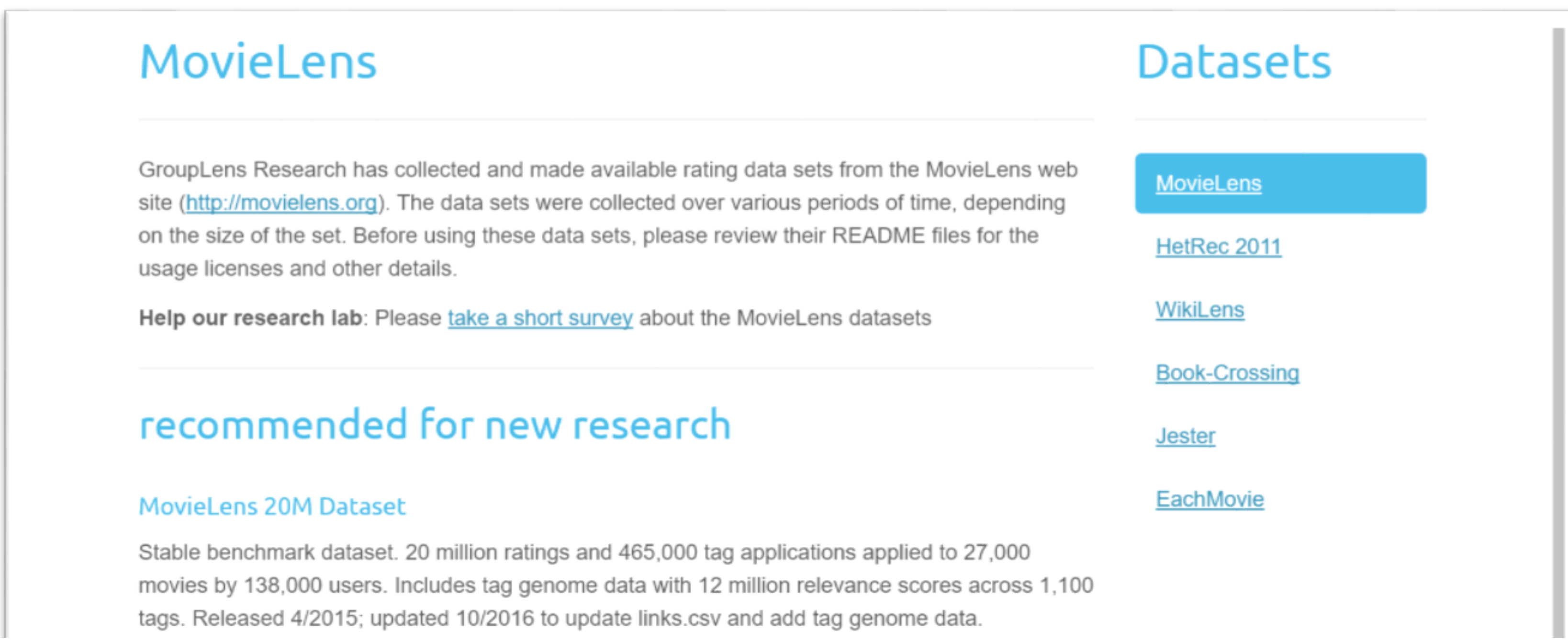
- The *k*-th pass
  1. Generate candidate items of size *k* from the frequent itemsets of size *k*-1
  2. Count for the candidate items
  3. Select only frequent itemsets of size *k*
  4. If there is a selected itemset, move on to the next pass;

     otherwise, stop.



Practice

- Construct a movie recommendation systems based on a market-basket analysis
  - codebase: https://github.com/shinhong/MovieLens
  - data: a part of the MovieLens dataset (https://grouplens.org/datasets/movielens/)



MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Help our research lab: Please take a short survey about the MovieLens datasets

recommended for new research

MovieLens 20M Dataset

Stable benchmark dataset. 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Released 4/2015; updated 10/2016 to update links.csv and add tag genome data.

Datasets

- MovieLens
- HetRec 2011
- WikiLens
- Book-Crossing
- Jester
- EachMovie

Design

- Basket
  - A set of movies whose rating by a user is greater than or equal to 4.0 out of 5.0 (i.e., the user likes the movies)
  - Around 2000 baskets are given for training.

- Recommendation problem
  - For a fresh user, the set of movies that the user likes is given as condition.
  - Predict whether the user would like a certain movie or not for the given condition.

Team

Team1	김보희	유진주	
Team2	권예성	추유진	
Team3	전영민	조정인	
Team4	이주영	김효림	
Team5	백건호	윤지영	
Team6	김효서	이찬혁	
Team7	김범준	윤한규	
Team8	안동민	박건희	
Team9	이예준	박예겸	
Team10	최지우	박지현	
Team11	정현섭	김정환	
Team12	이재익	강예찬	
Team13	고언약	이청준	
Team14	김빛나	감제원	조한나
Team15	차은비	김지석	정겨운
Team16	최시령	오재영	김다은
Team17	윤재원	강준민	김시민
Team18	유기쁨	김세인	김소은
Team19	김진향	김재빈	하재경
Team20	이용호	김상화	박지은

Homework

- Task1
  - Download the codebase from https://github.com/shinhong/MovieLens
  - Complete the missing parts (i.e., TODO's)
    - `showRatingStat()` of `MovieRatingData`
      - Draw a histogram that shows the distribution of the ratings in the given data
    - `predictPair()` method of `Recommender`
    - Constructor and `compareTo()` of `IntTriple` class
- Task2. Find at least 3 rules that have high confidence values
  - Describe at least 3 rules with actual movie titles
- Task3. write at least one idea of improving the recommendation system
  - Improve precision and accuracy?
  - Improve efficiency and scalability?
  - Improve robustness?
- Submission – deadline: 5:30PM, 11 Oct 2017
  - Source code for Task 1
  - Text file for Tasks 2 and 3

Discussion

- How to improve precision and accuracy?
- How to improve efficiency and scalability?
- How to improve robustness?