Neighborhood-restricted mining and selection of association rules for recommender systems

Fatih Gedikli and Dietmar Jannach

TU Dortmund, Germany

dietmar.jannach@tu-dortmund.de

Background

- Collaborative filtering recommender systems
  - Recommendation of items based on community behavior
- Given:
  - Users and item rating matrix
- Predict:
  - Ratings for unseen items for active user

Association rule mining (ARM)

- Designed to discovery buying patterns in sales transactions
  - Apriori algorithm (1994)
    - One of the earliest, more efficient algorithms

cheese ➔ beer [support = 10%, confidence=80%]

- Quality of rules
  - *Support*: How many times did cheese and beer appear together in all transactions
  - *Confidence*: How many times did beer appear in a transaction, when cheese was also bought?

Recommending based on ARM

- Building a Top-N recommender

  (1) Use Apriori to detect association rules that surpass minimum support threshold

  (2) Take rules that are "supported" by the active user (user has purchased item on LHS)
  - Compute set of items recommended by these rules

  (3) Sort the list by rule confidence
  - Variations of the scheme possible

ARM-based recommendation

- Early successful experiments reported
  - Accuracy comparable to kNN-baseline
- Advantages of ARM-based recommendation
  - Offline model-building phase possible
  - Efficient query-time recommendation process
- Explanations for recommendations
  - Increasingly important as a quality aspect of RS
  - Particular importance of the aspect of “transparency”
- Incorporation of business rules
  - Possible due to the explicit knowledge model

Improvements to basic Apriori

- Limited coverage of basic Apriori
  - Reported, e.g., in Sandvig et al. 2007.
  - Problem of finding (retaining) rules for rare items because of global support threshold
- Recent improvements
  - Use individual support values per user or item (Lin et al. 2002); leads to slight accuracy improvements.
  - Kiran and Reddy (2009): IMSApriori
    - Use new metric to determine the minimum support values
    - Not evaluated yet in RS domain

In this talk

- Evaluation of IMS-Apriori for RS
  - Including Frequent Itemset Graph data structure
- Evaluation of new ARM-based RS algorithm NRR
- Basic idea:
    - Closer neighbors are better predictors than others
    - Learn **individual rule sets per user** based on a limited neighborhood
    - Utilize personal rule base plus the neighbor's rules for more accurate recommendations

Example

r1

Rare item

	Item 1	Item 2	Item 3	Item 4	...	Item 6	Item 7	Item 8
User 1	1	0	1	0	...	?	?	?
User 2	1	0	1	0	...	1		
User 3	1	0	1	0	...	1		
User 4	1	0	1	1	...		1	1
User 5	1	0	1	1	...			1
User 6	1	1	1	1	...			1
...	...	...	...	...	...	...	...	...

r2

- Goal is to recommend item to User1
  - Apriori might find these rules (and recommend Item8)

$$\text{r1: } Item1, Item3 \Rightarrow Item6$$
$$[\ support = 33\%, confidence = 33\%\ ] \text{ and}$$
$$\text{r2: } Item1 \Rightarrow Item8$$
$$[\ support = 50\%, confidence = 50\%\ ].$$

Example ctd.

r1

Rare item

	Item 1	Item 2	Item 3	Item 4	...	Item 6	Item 7	Item 8
User 1	1	0	1	0	...	?	?	?
User 2	1	0	1	0	...	1		
User 3	1	0	1	0	...	1		
User 4	1	0	1	1	...		1	1
User 5	1	0	1	1	...			1
User 6	1	1	1	1	...			1
...	...	...	...	...	...	...	...	...

r2

- Observations
  - User2 and User3 are “closer” to User1
  - And might thus be better predictors for User1

- Thus, we propose to learn rules using only the transactions of the neighbors
  - rule “Item1 ➔ Item6“ will thus get more importance.
  - Coverage can be re-increased by using rules of User4 also for predictions

Algorithm sketch

In: *user, ratings, learnNS, predictNS,* λ, α

(1) Calculate frequent itemsets per user

(2) Determine prediction neighborhood of user

(3) Compute recommendations for every user in the neighborhood (based on rule confidence)

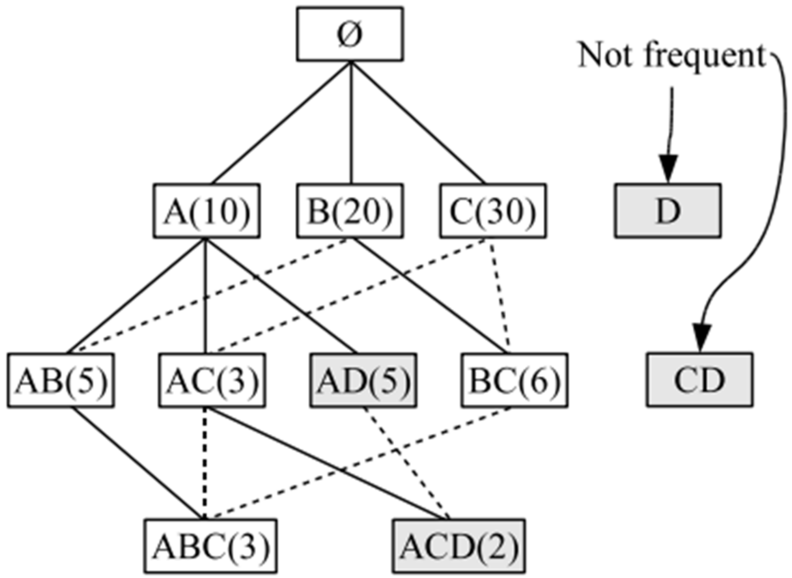
(4) Calculate weighted combination of own and neighbor recommendations based on user similarity

The Extended Frequent Itemset Graph

- Data structure proposed in Mobasher et al. 2001
  - Make recommendations based on frequent itemsets without calculating association rules
  - Works on the assumption that

    "Every subset of a frequent itemset is also a frequent itemset"
  - Assumption does not hold if multiple support values are used
  - Extended FIG proposed in our work
    - Add additional starting points for tree-traversal

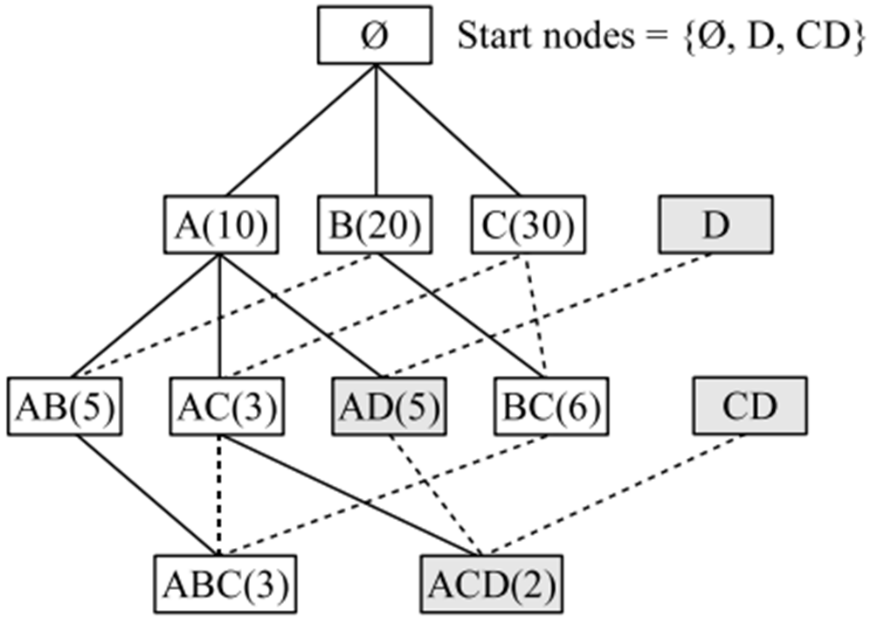
EFIG



- Standard method:
  - Past user transaction {AD}
  - Search for superset of {AD}, e.g., {ACD}
  - Recommend C based on support values
  - {D} and {CD} are however **not** frequent
    - Although they are subsets of {AD} and {ACD}
    - Cannot recommend A for users that purchased {CD}, although this would be plausible

EFIG (2)

- Determine subsets of frequent itemsets
  - e.g., {D} and {CD}
- Add these nodes as additional start nodes for tree traversal
- Connect the nodes to their superset nodes



- Restart depth-first traversal on (small) subset of graph from these additional nodes
- Only small part of the tree will be re-visited

Experimental evaluation

- Two datasets
  - MovieLens 100k
    - 943 users, 1.682 items; only users with at least 20 ratings
  - Yahoo!Movies
    - 211.000 ratings, 7.600 users, 12.000 items; at least 10 ratings per user
- Variation of density level (10% to 90%)
  - Low-density levels typical in real-world data sets
  - Four-fold cross-validation

Accuracy metrics

- Like statements and ratings
    - Determine "like" ratings from Likert scale ratings
    - Like = Item is rated above the user's average
- Precision and Recall
    - Retrieve top-N recommendation list based on training set
    - Compare set of existing like statements (ELS) in test set with set of predicted like statements (PLS)

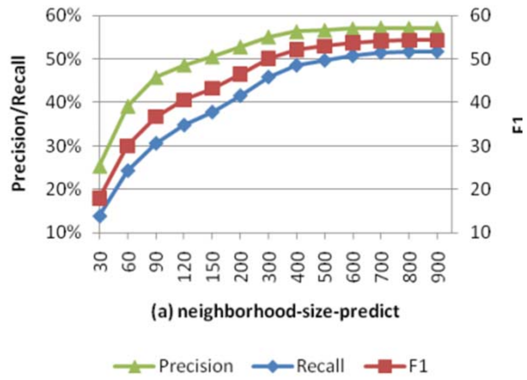
– Precision = $(|PLS| \cap |ELS|) / |PLS|$

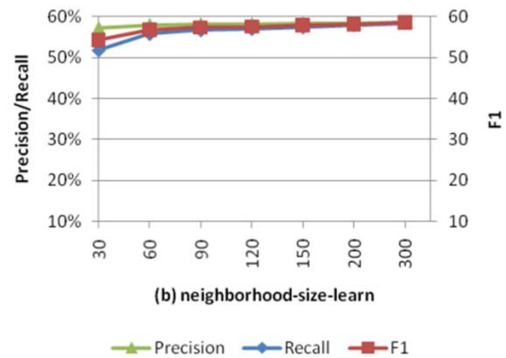
– Recall = $(|PLS| \cap |ELS|) / |ELS|$

– F1: Harmonic mean of Precision and Recall

Algorithm parameters

- kNN
  - Use default voting and neighborhood size = 30
  - Based on literature
- Neighborhood-sizes for NRR
  - Based on sensitivity analysis
  - Learning, 60 neighbors
  - Predicting, 900 neighbors
- Support threshold LS
  - Difficult task
  - Empirical values:
    - IMSApriori: 3%, NRR: 9%





Offline phase

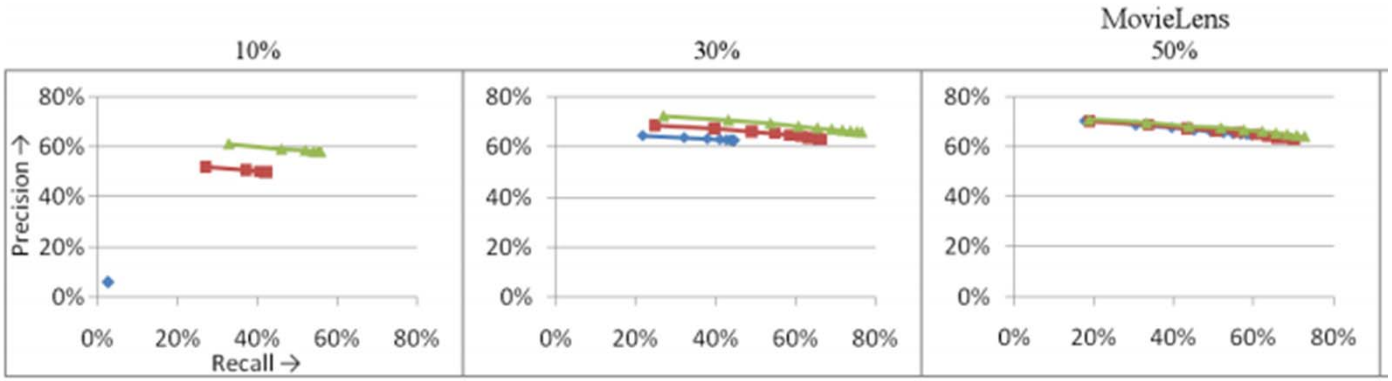
- Model-learning times and model sizes
  – Learning all rule bases for MovieLens 40%
    - 8 minutes on a standard desktop PC
    - Average model size is 49 rules for this setting
  – IMSApriori: 182 rules learned
    - Depends of course on the chosen LS value

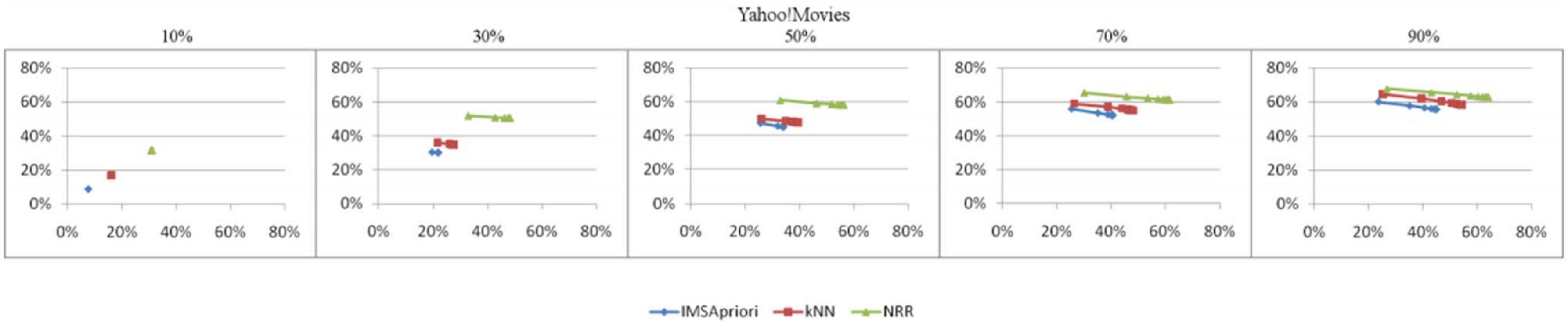
Detailed results (Top 10)

		Density ➔	10%	20%	30%	40%	50%	60%	70%	80%	90%
MovieLens	F1	kNN	45,83	56,68	64,65	65,52	66,26	66,16	65,74	65,4	64,79
		IMSApriori	3,79	33,31	52,22	59,08	62,06	63,06	63,54	62,9	62,31
		NRR	56,82	68,2	70,78	70,38	68,12	67,45	66,53	65,3	63,79
	Precision	kNN	49,89%	58,10%	63,02%	62,52%	62,85%	63,27%	63,82%	64,30%	64,59%
		IMSApriori	5,94%	48,25%	62,63%	64,53%	64,76%	65,10%	65,48%	65,35%	64,75%
		NRR	57,92%	64,00%	65,90%	65,30%	64,11%	64,43%	64,30%	64,00%	63,09%
	Recall	kNN	42,37%	55,33%	66,38%	68,82%	70,06%	69,33%	67,78%	66,54%	65,00%
		IMSApriori	2,80%	25,45%	44,78%	54,49%	59,59%	61,16%	61,71%	60,63%	60,05%
		NRR	55,75%	73,00%	76,45%	76,31%	72,66%	70,77%	68,93%	66,66%	64,51%
Yahoo!Movies	F1	kNN	13,97	22,64	30,69	37,41	43,2	47,61	51,37	53,94	56,24
		IMSApriori	6,95	17,11	25,19	32,43	38,86	43,17	45,79	48,05	49,78
		NRR	29,16	42,22	49,4	52,66	57,15	59,84	61,39	62,23	63,51
	Precision	kNN	15,95%	26,11%	34,85%	42,12%	47,60%	51,79%	54,82%	56,61%	58,25%
		IMSApriori	7,64%	20,15%	29,98%	38,86%	45,15%	49,77%	52,27%	54,74%	55,86%
		NRR	30,67%	43,43%	50,54%	54,14%	58,02%	60,21%	61,22%	61,82%	62,77%
	Recall	kNN	12,43%	19,99%	27,42%	33,65%	39,54%	44,06%	48,32%	51,51%	54,37%
		IMSApriori	6,37%	14,88%	21,73%	27,82%	34,10%	38,12%	40,75%	42,82%	44,90%
		NRR	27,78%	41,07%	48,31%	51,26%	56,31%	59,48%	61,57%	62,65%	64,27%

Figure 4: Top 10 F1, precision and recall values for different density levels.

Varying recommendation list lengths





Summary & Future work

- Accuracy improvement in both data sets
  - Significant improvements on low density levels
  - Improvements stronger for the sparser Yahoo!Movies data set
- Advantages of ARM-based approaches
  - Scalability, robustness against attacks, explanations, incorporation of business rules
- Future work
  - Further data sets
  - Experiments with clustering approaches
  - Taking “dislike” statements into account

Discussion

- Questions?
- Mixing in business rules in CF approaches?
- Repeatability of experiments
  - Algorithms and data can be accessed