Overlay Management for Fully Distributed User-based Collaborative Filtering

Róbert Ormándi, István Hegedűs, Márk Jelasity

University of Szeged
2010

Recommender system - background

- Recommends items (music, movie, book) for users based on their preferences (ratings)
- There are many web shops (e.g. Amazon) that use these type of algorithms
- You rate some things, that you like/dislike and the system recommends „other good things” for you
- Common approaches is the user based Collaborative Filtering methods

Collaborative Filtering (CF) - background

- Needs a correlation or similarity measurement between the users
- The recommendation based on the weighted summarized ratings, using this function:

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_u} s_{u,v} \left(r_{v,i} - \bar{r}_v\right)}{\sum_{v \in N_u} \left|s_{u,v}\right|} + \bar{r}_u$$

  - r: rating
  - s: similarity

Evaluation of recommender systems - background

- There are some manually labeled benchmark datasets (e.g. BookCrossing, MovieLens, …)
  - One part of these databases (train) are for fine tune the parameters (learning). E.g. building the overlay
  - The remaining part for test or evaluation (computing the differences between the expected and the predicted votes)
- The commonly used evaluation metric is the MAE (Mean Absolute Error)

Decentralization

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_u} s_{u,v} \left(r_{v,i} - \bar{r}_v \right)}{\sum_{v \in N_u} \left| s_{u,v} \right|} + \bar{r}_u$$

- Centralized case
  - Available the full dataset
  - precise, need power servers, storage devices
  - N_u: is the set of all users
- Decentralized and Distributed (P2P) case
  - N_u: the neighbor set of the user u (with size k)
  - Find the most relevant users → manage an overlay network
  - Too many neighbors can make pretty much load

Our task

- Make an overlay management service which
  - Supports CF method
  - Close to the optimal recommendation in term of the performance and the load of the network (trade-off)
- There is no this type of comparison of distributed recommender systems

Dataset properties

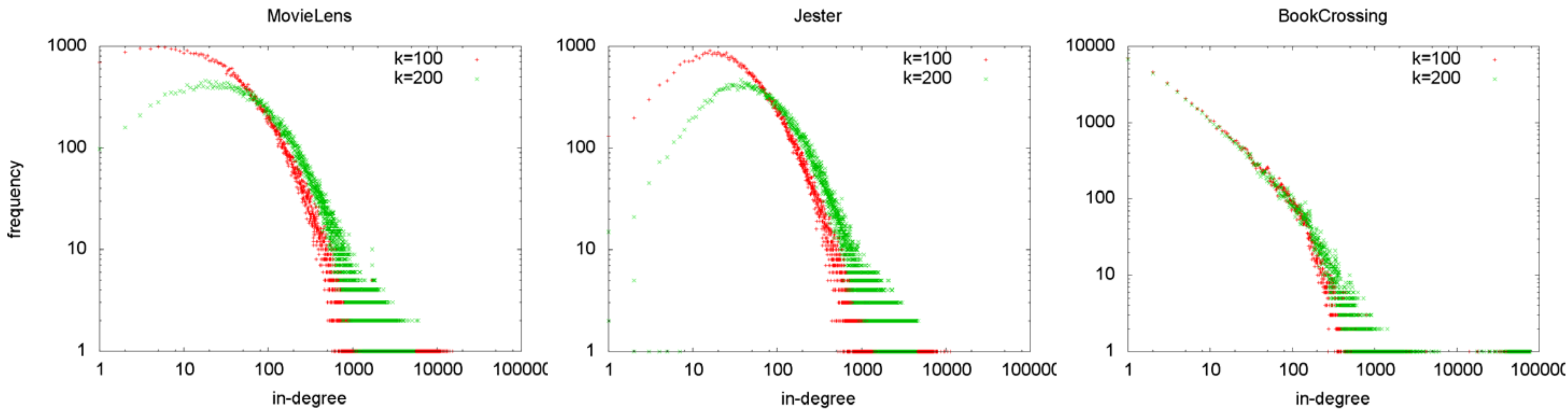
- Base statistics and properties of the three used recommender datasets:

	MovieLens	Jester	BookCrossing
# users	71,567	73,421	77,806
# items	10,681	100	185,974
size of train	9,301,274	3,695,834	397,011
sparsity	1.2168%	50.3376%	0.0027%
size of eval	698,780	440,526	36,660
eval/train	7.5127%	11.9195%	9.2340%
# items $\geq$	20	15	1
rate set	$1, \ldots, 5$	$-10, \ldots, 10$	$1, \ldots, 10$
MAE(med)	0.93948	4.52645	2.43277

  – The train/test cutting based on rating occurrences by users

Dataset properties (2)

- Power-law in-degree distribution by the perfect kNN overlay network



- It makes too much load

P2P overlay management algorithms

- The algorithms build and manage the user-similarity based overlay
- On the top of this overlay works a user-based CF algorithm
- → we focus on overlay management
- We use the earlier mentioned aggregation method, and the Cosine similarity measure
- We would like to keep the load low

P2P overlay management algorithms (2)

- Our basic algorithm

Algorithm 1 Random Nodes based Overlay Management

Parameters: k: the size of view; r: the number of randomly generated nodes

while true **do**
    $samples \leftarrow$ getRandomPeers(r)
    for $i = 1$ **to** r **do**
        $peer \leftarrow$ get($samples, i$)
        $peerDescriptor \leftarrow$ descriptor($peer$)
        insert($view, peerDescriptor$)

  - view: a bounded priority queue for the neighbors and contains descriptors
  - random peer selection from the network by the NewsCast

P2P overlay management algorithms (3)

- BuddyCast (baseline)
  - Buddy, candidate, stop, random lists
- kNN graph from random samples
  - Random node insertion into view list
- kNN graph by T-Man (merge view lists)
  - Global
  - View
  - Proportional
  - Best

T-Man based algorithms

- Global: randomly selected peer for the communication by the NewsCast from the whole network
- View: uniformly selected peer from the view of the current node
- Proportion: like the View, but we defined the probability distribution as:

$$p_{i,j} = \frac{\frac{1}{sel_j+1}}{\sum_{k \in View_i} \frac{1}{sel_k+1}}$$

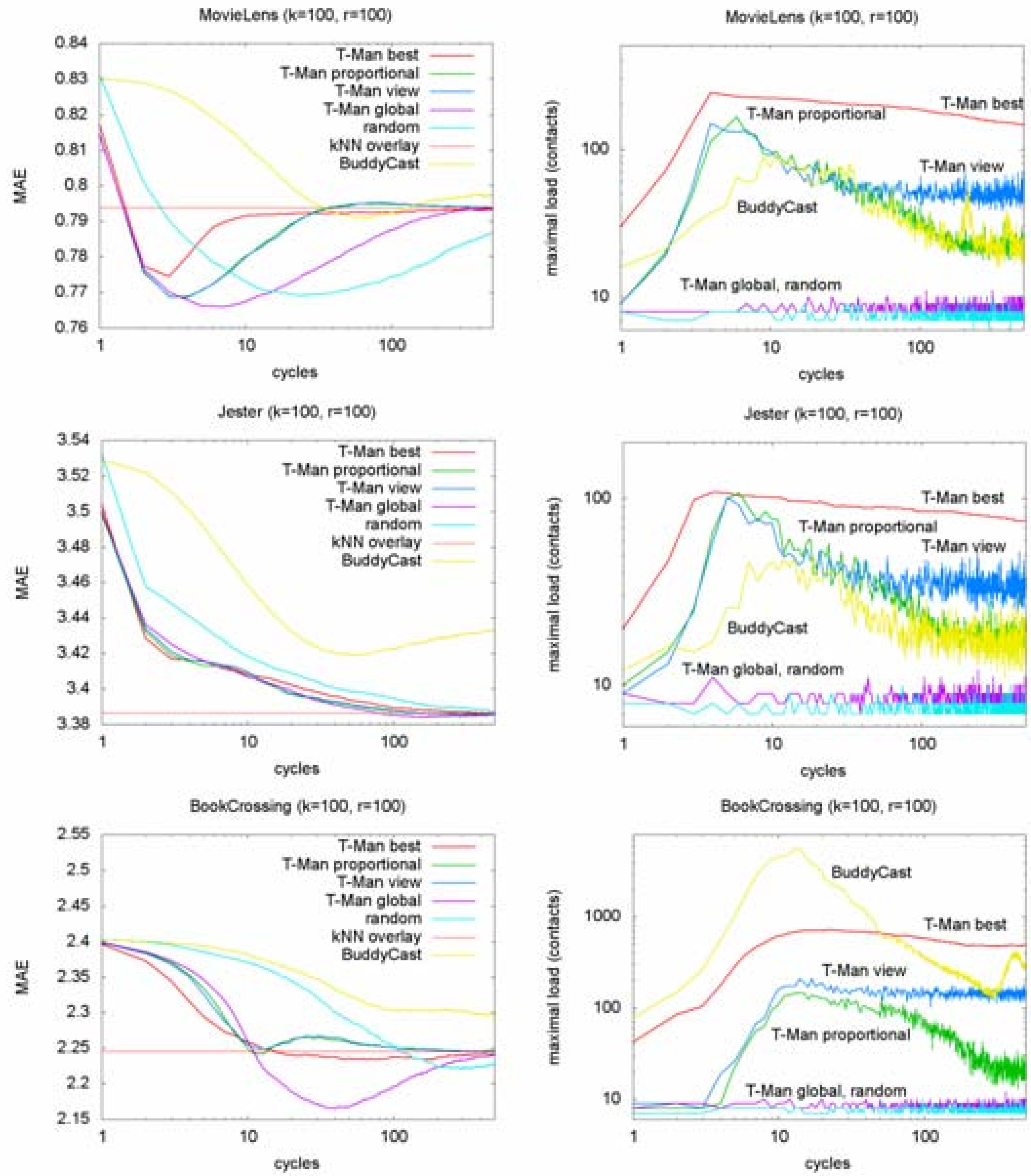
- Best: we selected the most similar node from the view for the communication

Results (algorithm settings)

- BuddyCast:
  - Buddy list size: 100
  - Candidate list size: 100
  - Random list size: 10
  - Block list size: 100
  - Exploration factor: 0.5
- Our algorithms:
  - View list size: 100
  - Random peer selection size: 100

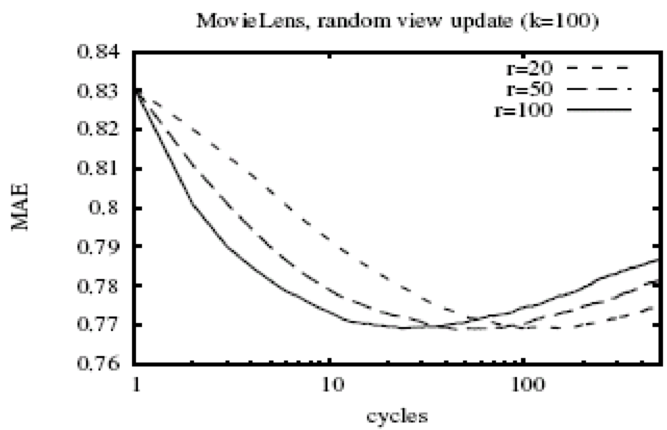
Overall results

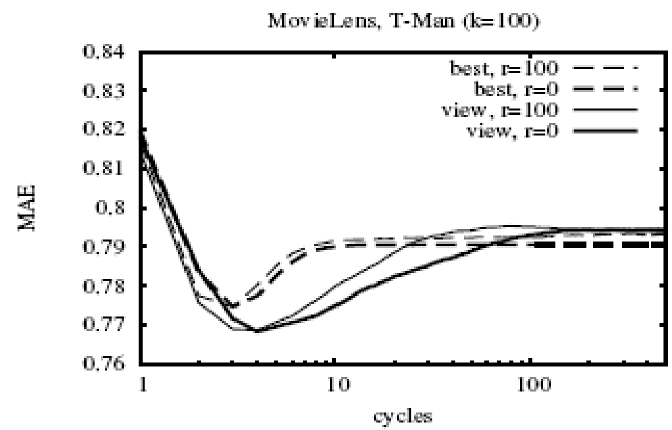
- The performance of the algorithms measured in MAE (mean absolute error), and the cost of the load
- The x-axis represents the convergence speed measured in cycles



Results (the r parameter)

- Given some random samples from the network to
  - Avoid the local optima
  - Increase the convergence speed





- The effect of r on the other databases and settings is similar

Keep the minimum

- If we know the cycle number (c) of the minimum, we can keep the algorithm at this point. Just choose the top k similar peer from the c*r random samples (it does not make extra load). ☺