Recommendation Systems: Concepts, Techniques, and Applications

2019. 06. 26

Sang-Wook Kim

Department of Computer Science and Engineering
Hanyang University

CSE Department @ Hanyang University

Hanyang Univ.

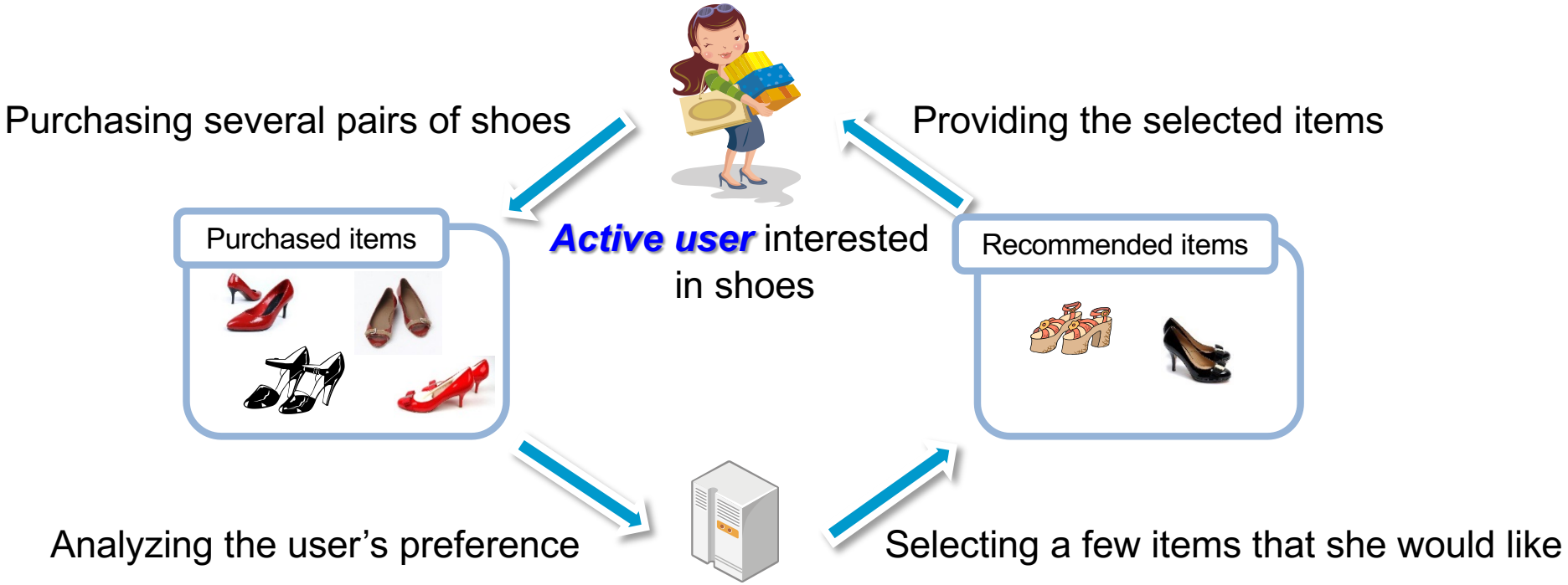
- Hanyang University
  - Established in 1939
  - Located in Seoul, Korea
- Department of Computer Science and Engineering
  - Members
    - Around 600 undergraduate students
    - More than 250 graduate students
    - 25 faculty members: Cover all the key areas in computer science
  - Good research environment and facilities in our graduate program
  - Various scholarships by the funding from government and industry
  - Visit our web site at http://www.hanyang.ac.kr

Contents

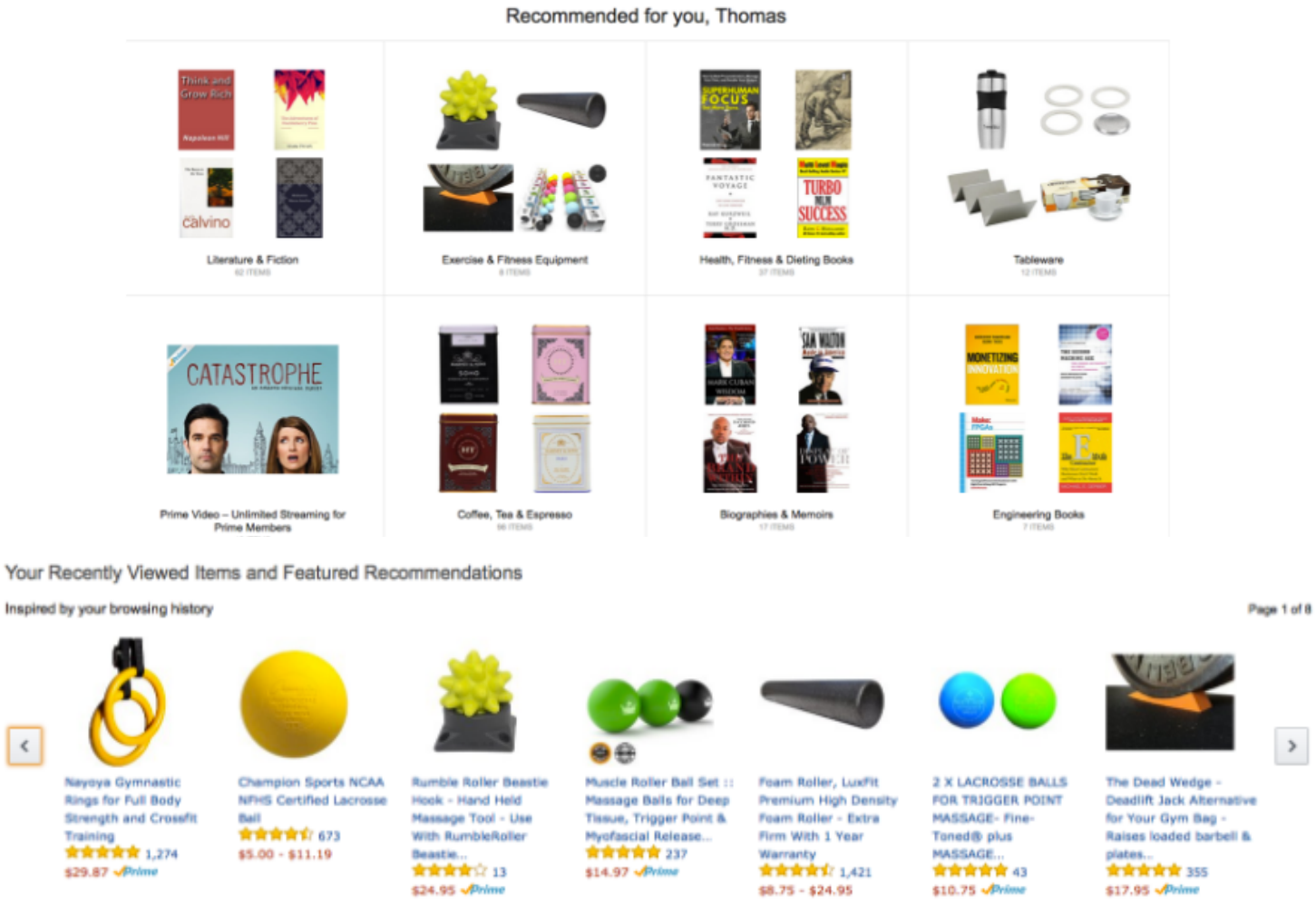
- **Recommendation Systems: An Overview**
  - Concepts and applications
  - Collaborative filtering (CF)
- Exploiting Uninteresting Items for Effective CF
  - Ratings setting
  - One-class setting
- Some More Research Results
  - Recommendations
  - Graph engines
- Conclusions

Recommendation Systems

- Provide a user with a few items that she would like
  - Using her history of evaluating, purchasing, and browsing



Amazon



http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/

Netflix

https://www.netflix.com/

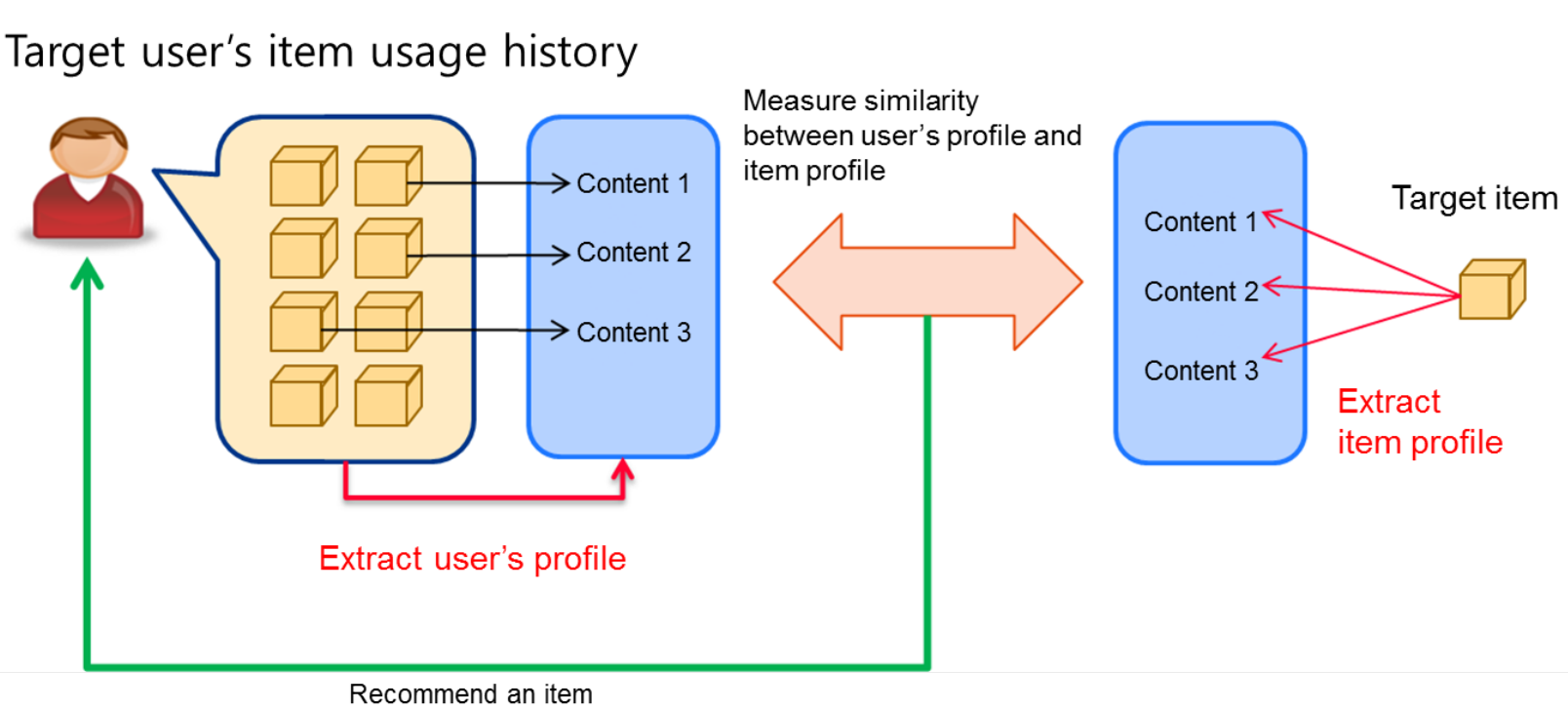
Many Applications

Classification of Recommendation Systems

- Content-based approach
  - Recommending those items that have similar contents to those of the active user's favorite items
- **Collaborative filtering (CF) approach (our focus)**
  - Recommending items rated high by neighbors who have preferences similar to that of the active user
- Trust-based approach
  - Recommending items based on trust relationships among users
- Hybrid approach
  - Recommending items by combining the approaches above

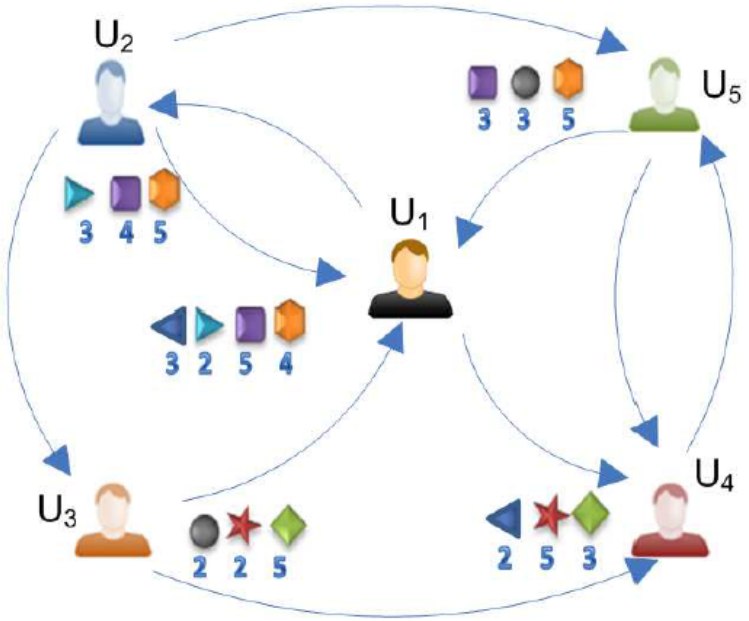
Content-Based Approach

- Recommend such items that have contents similar to the profile of the active user



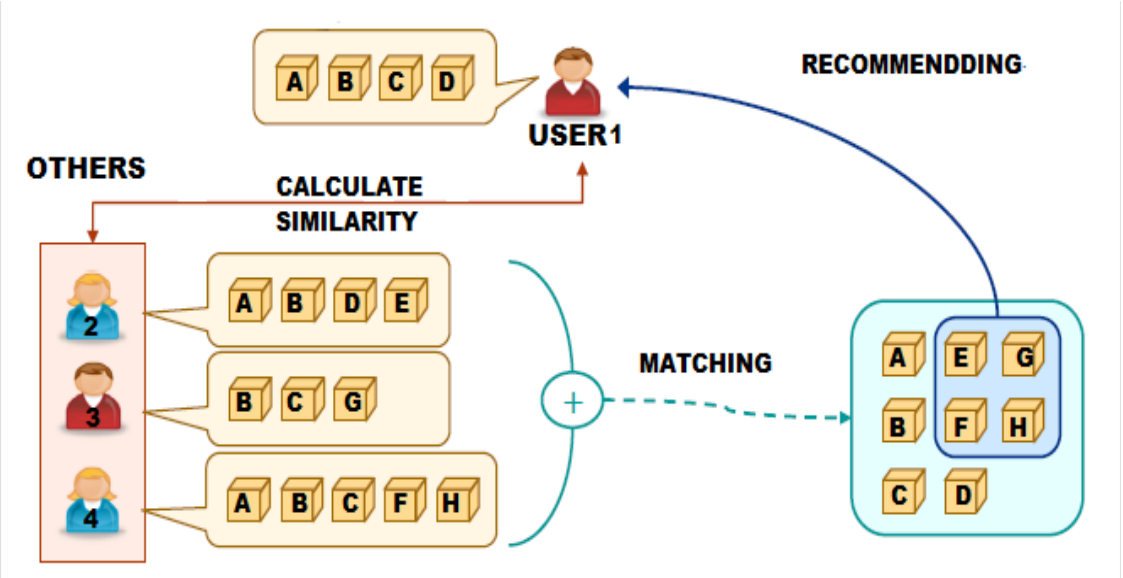
Trust-Based Approach

- Recommend such items rated high by trustable users who have direct or indirect trust relationships with the active user



Collaborative Filtering (CF) Approach

- Recommend such items rated high by neighbors who have preferences similar to that of the active user
  - Step 1: *Finding a group of users* (*neighbors*) whose preferences are similar to that of an active user *c*
  - Step 2: *Estimating $r_{c,s}$, the rating of item s for active user c,* based on the ratings given to item *s* by *c*'s neighbors
  - Step 3: Recommending *a few items with the ratings estimated high*



Collaborative Filtering (CF)

- Data: rating matrix

Items

Users

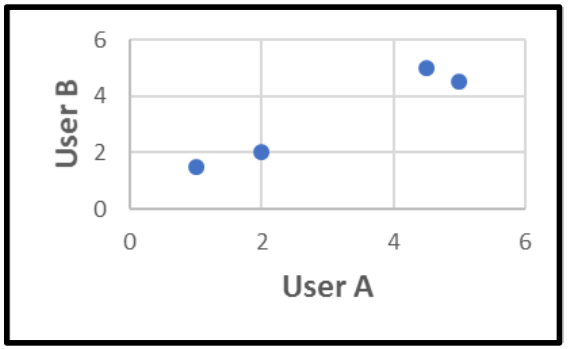
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1		1		1		5	3			5		
u_2	1	3		4		4				2	4	
u_3												5
u_4		3		4		4		4	3			
u_5			4									
u_6	1			3	5	1		4			3	

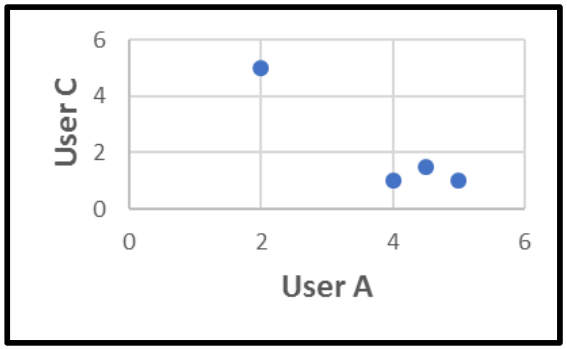
A rating on i_1 given by user u_6

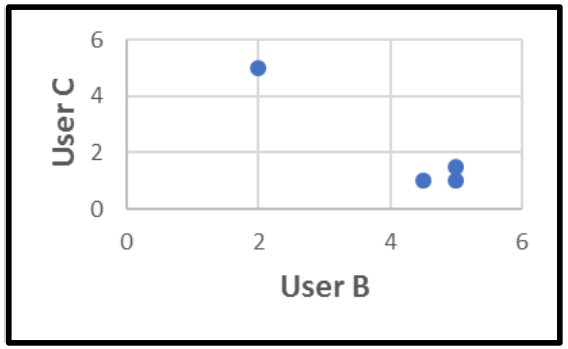
Heuristic-based Method in CF: Step 1

- Similarity measure for users a and i (used in finding neighbors)
  - Example: three users' ratings for six movies
    - User A = <4.0, 1.0, 4.5, 5.0, 2.0, _>
    - User B = < _, 1.5, 5.0, 4.5, 2.0, 5.0>
    - User C = <1.0, _, 1.5, 1.0, 5.0, 1.0>
  - Pearson correlation coefficient (PCC)

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$



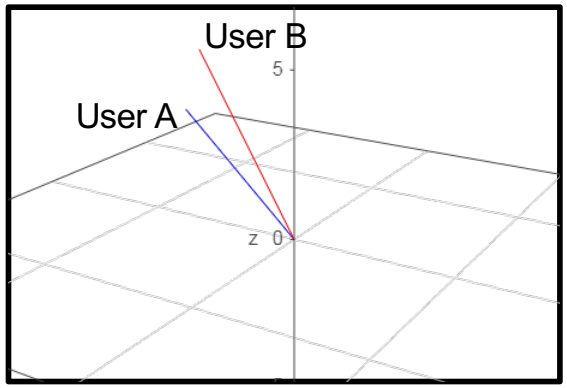


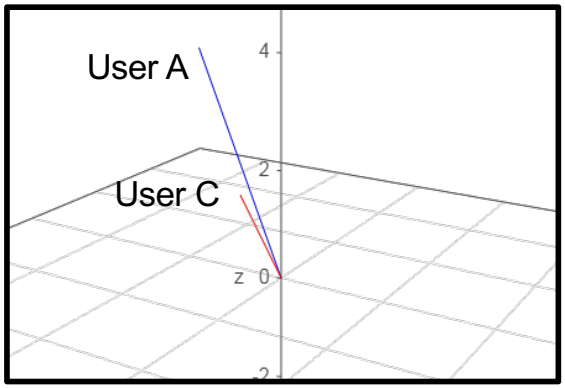


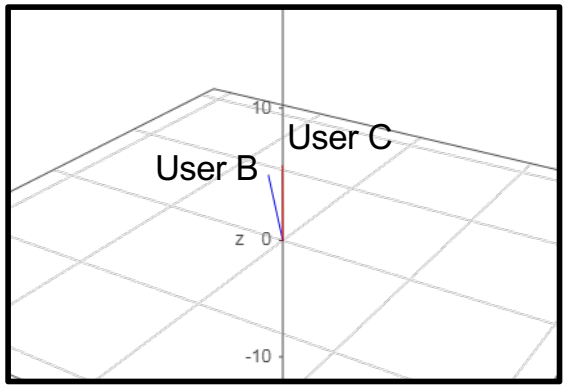
Heuristic-based Method in CF: Step 1

- Similarity measure for users a and i (used in finding neighbors)
  - Example: three users' ratings for six movies
    - User A = <4.0, 1.0, 4.5, 5.0, 2.0, _>
    - User B = < _, 1.5, 5.0, 4.5, 2.0, 5.0>
    - User C = <1.0, _, 1.5, 1.0, 5.0, 1.0>
  - Cosine similarity

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$







Heuristic-based Method in CF: Step 2

- Aggregation of ratings on a target item given by the neighbors

$$r_{c,s} = \underset{c' \in \hat{C}}{\text{aggr}}\, r_{c',s}$$

  - $r_{c,s}$: Estimated rating on item s for user c
  - $\hat{C}$: Set of neighbors for user c
  - Different methods for aggregation

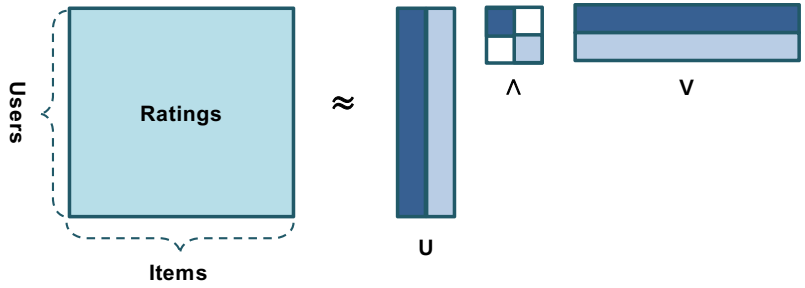
$$(a)\ r_{c,s} = \frac{1}{N} \sum_{c' \in \hat{C}} r_{c',s}$$

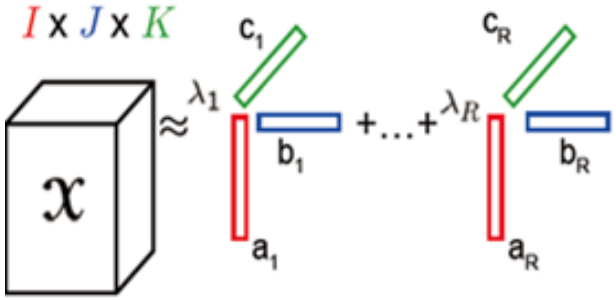
$$(b)\ r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \times r_{c',s}$$

$$(c)\ r_{c,s} = \bar{r}_c + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r}_{c'})$$

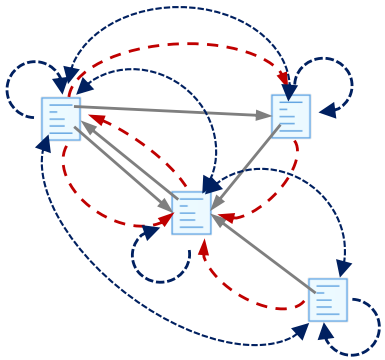
Machine-Learning Based CF Techniques

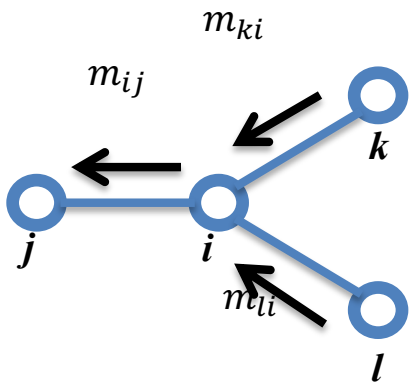
- Matrix/Tensor Factorization



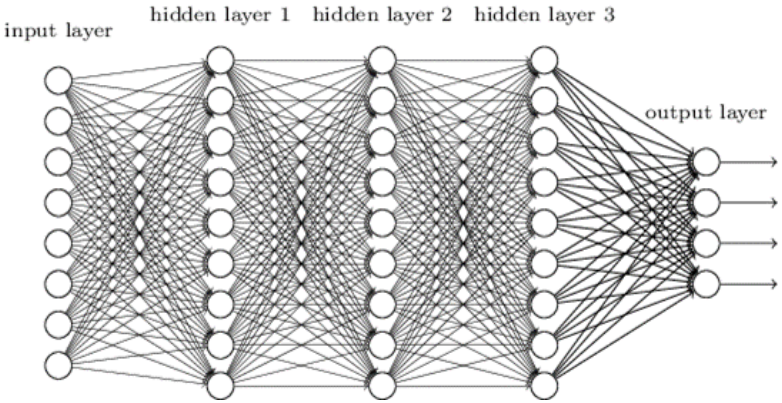


- Social Network Analysis

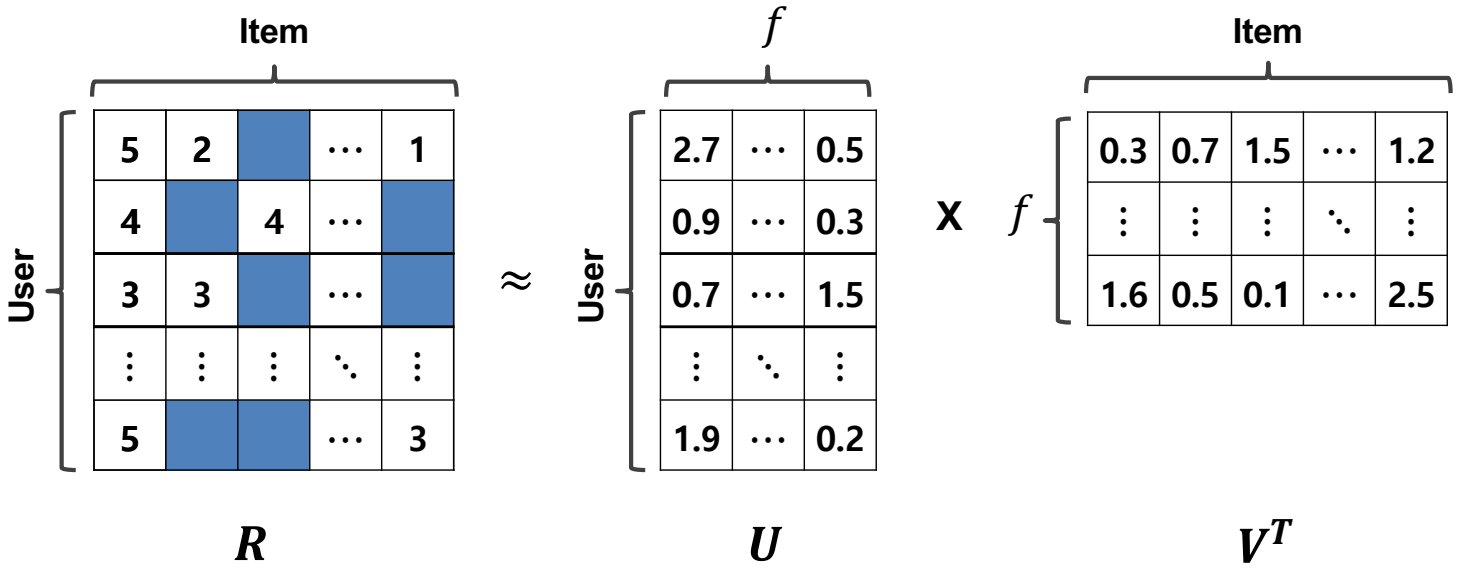




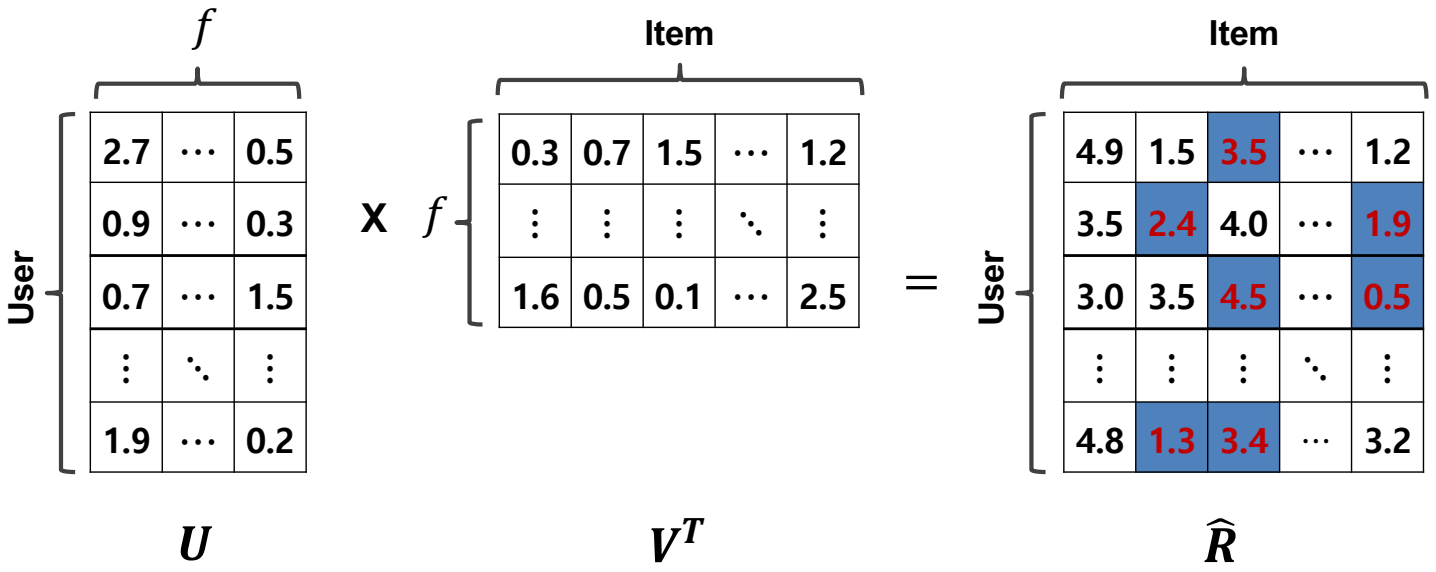
- Deep Learning



Matrix Factorization (MF)

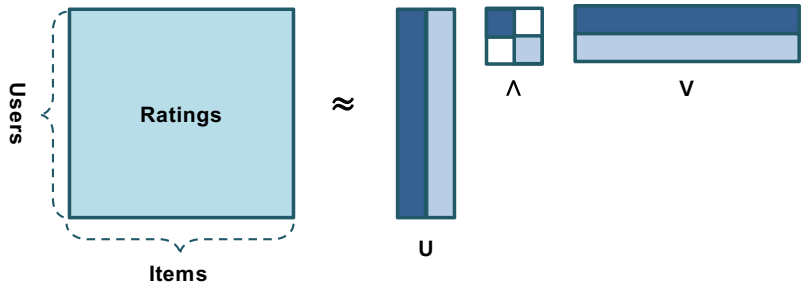


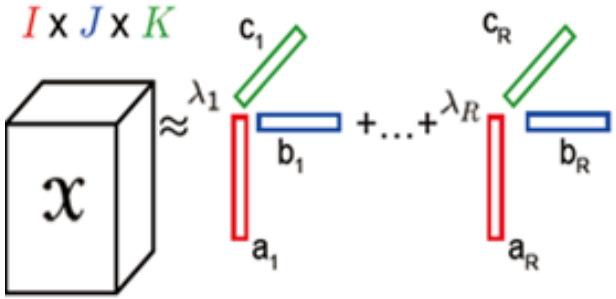
Matrix Factorization (MF)



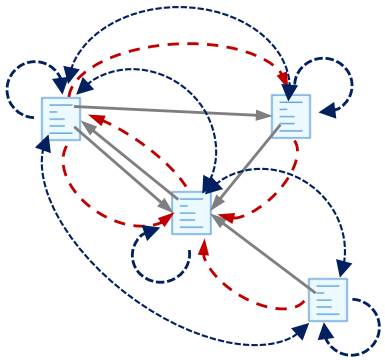
Machine-Learning Based CF Techniques

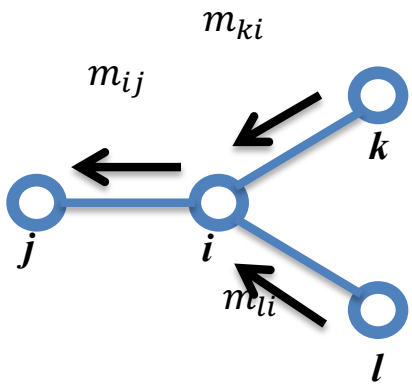
- Matrix/Tensor Factorization



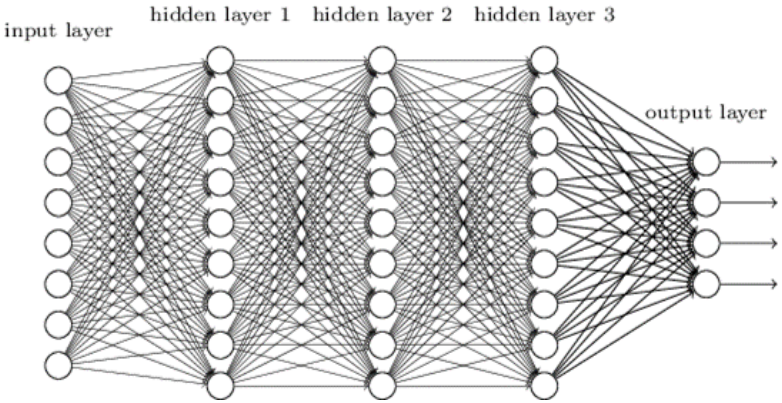


- Social Network Analysis





- Deep Learning



Contents

- Recommendation Systems: An Overview
  - Concepts and applications
  - Collaborative filtering (CF)
- **Exploiting Uninteresting Items for Effective CF**
  - Ratings setting
  - One-class setting
- Some More Research Results
  - Recommendations
  - Graph engines
- Conclusions

“Told You I Didn’t Like It”: Exploiting Uninteresting Items for Effective Collaborative Filtering

(IEEE ICDE’16 / IEEE TKDE’19)

Motivation

- CF approaches focus on *only the ratings given by users*
  - Data sparsity problem: most users have evaluated *only a few items*
  - There are only a few ratings in a rating matrix (< 4%)
- CF approaches suffer from *low accuracy and coverage*

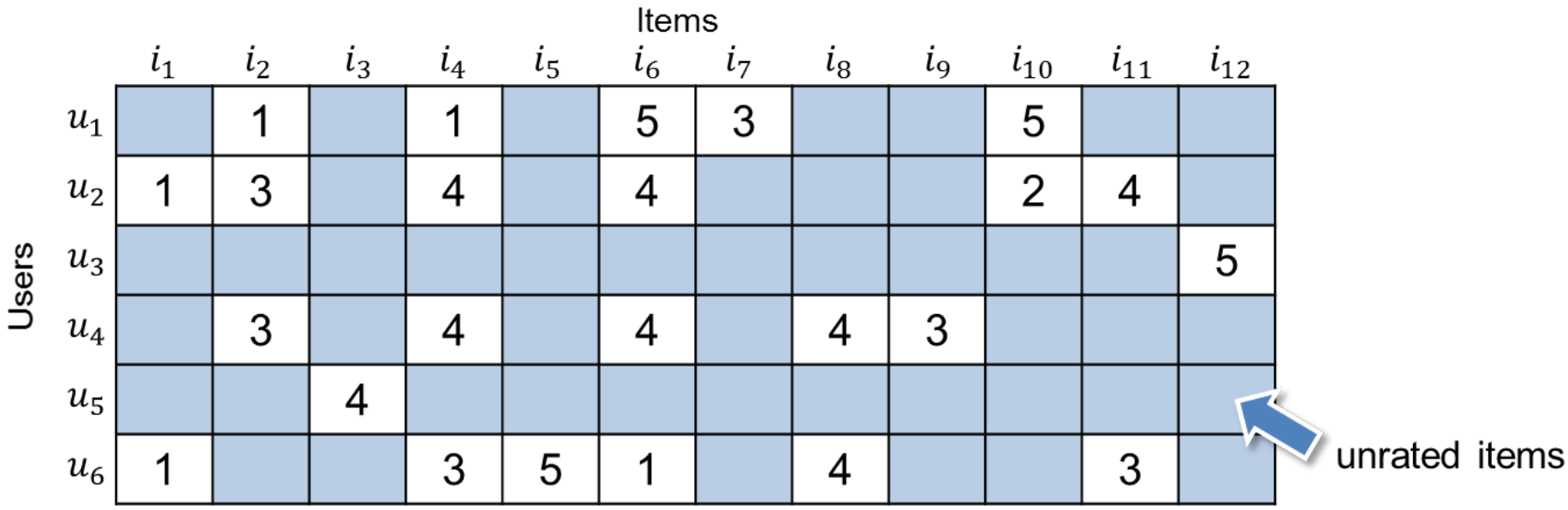
Users \ Items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1		1		1		5	3			5		
u_2	1	3		4		4				2	4	
u_3												5
u_4		3		4		4		4	3			
u_5			4									
u_6	1			3	5	1		4			3	

A rating on i_1 given by user u_6

A Rating Matrix

Our Idea: Using Unrated Items

- Exploit uncharted unrated items
  - A fraction of rated items in a rating matrix R is extremely small (< 4%)
  - Exploiting the vast number of "unrated" items in R can lead to a significant improvement in CF approaches



Our Idea: Uninteresting Items

- Unrated items
  - Users were not aware of their existence
    - Candidates for recommendation
  - Users knew but did not like and thus did not purchase
    - ***Uninteresting items***
- Uninteresting items (of a user)
  - Items on which the user has “negative” preferences

Users' Preferences: Two New Notions

- **Pre-use preferences**
  - User's impression on items before purchasing and using them
  - Determined via meta data of items (known before purchasing)
- **Post-use preferences**
  - User's impression on items after purchasing and using them
  - Determined via real content of items (unknown before purchasing)

Pre-Use/Post-Use Preferences and Ratings

Hanyang Univ.

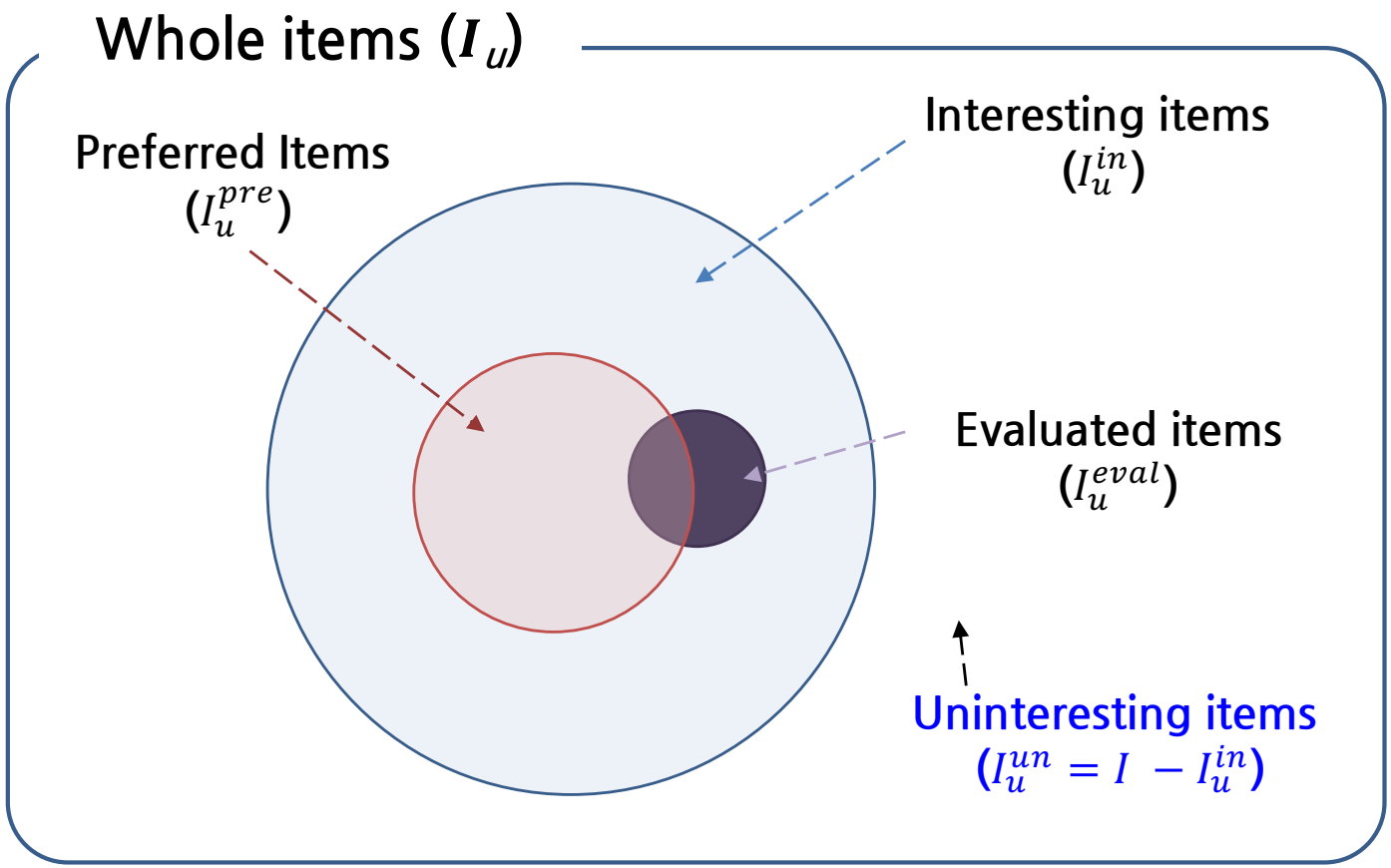
- A user has high pre-use preference for Movie #1 and Movie #2
  - The user likes Movie #1 but is disappointed at Movie #2
- A user has low pre-use preference for Movie #3

Movie	Pre-use preference	Post-use preference	Rating
Movie#1	High	High	5
Movie#2	High	Low	1
Movie#3	Not high	Unknown	Unrated

- In this case
  - Uninteresting items: Movie #3
  - Interesting items: Movie #1 (preferred) and Movie #2 (not preferred)

Venn Diagram for Preferences of Items

- Entire set of items that an active user thinks



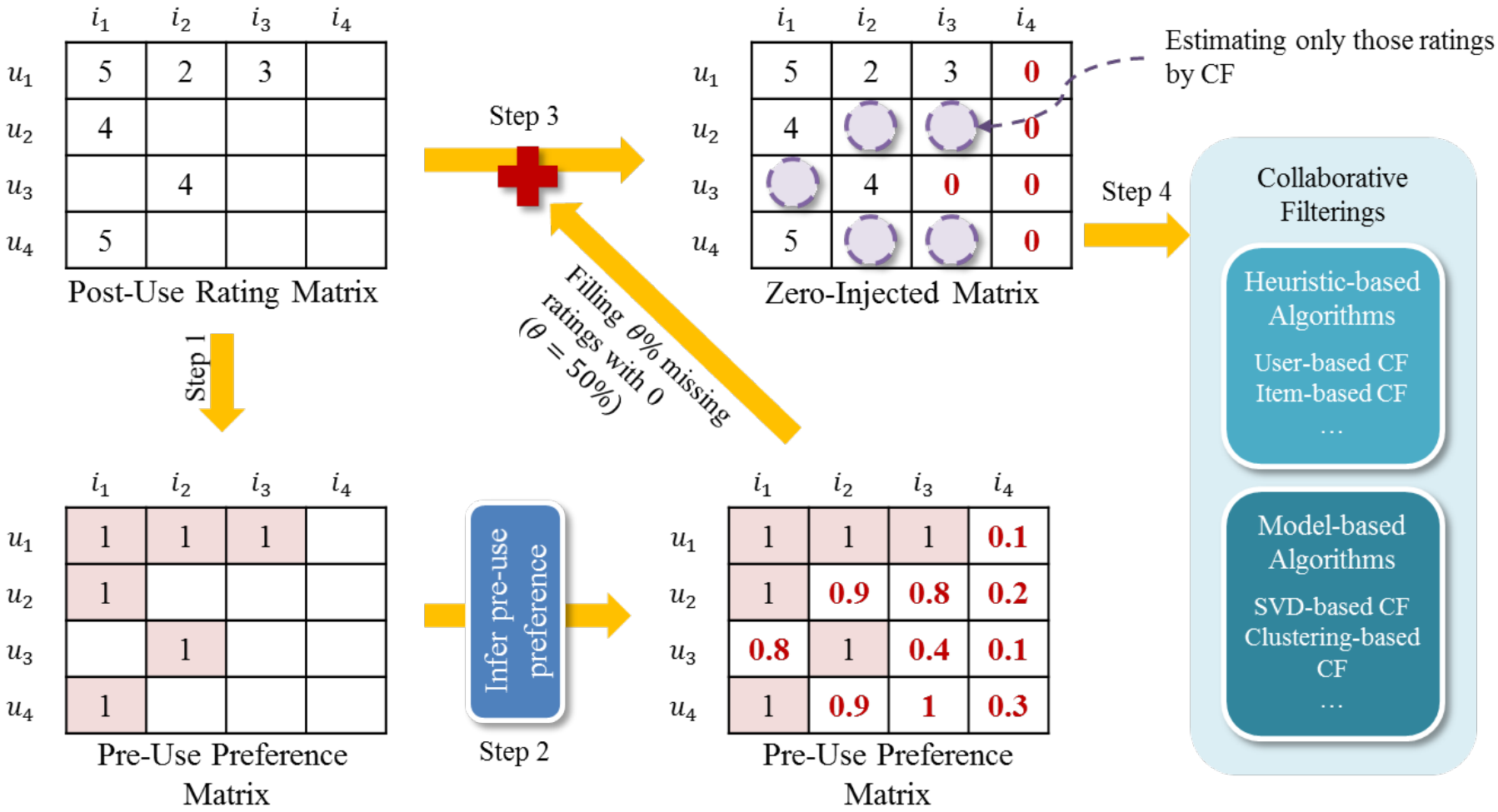
Pre-Use Preference and Uninteresting Items

- Challenge: To identify uninteresting items among unrated items
- A user's uninteresting items
  - Her pre-use preferences on them are relatively low
- How to know a user's pre-use preferences
  - For all "rated" items: highest pre-use preferences
    - Otherwise, users would not have bought them in the first place
  - For unrated items: pre-use preferences need to be inferred
    - Based on pre-use preferences on rated items

Exploiting Uninteresting items

- Our final goal
  - Identify top-*N* (preferred) items
    - They are interesting items (*i.e.*, with high pre-use preferences)
    - Their post-use preferences are higher than others
- Two strategies with uninteresting items
  - Strategy 1: to exclude uninteresting items from the final recommendation list
  - Strategy 2: to exploit both uninteresting items and rated items in CF

Zero-Injection (IEEE ICDE'16 / IEEE TKDE'19)

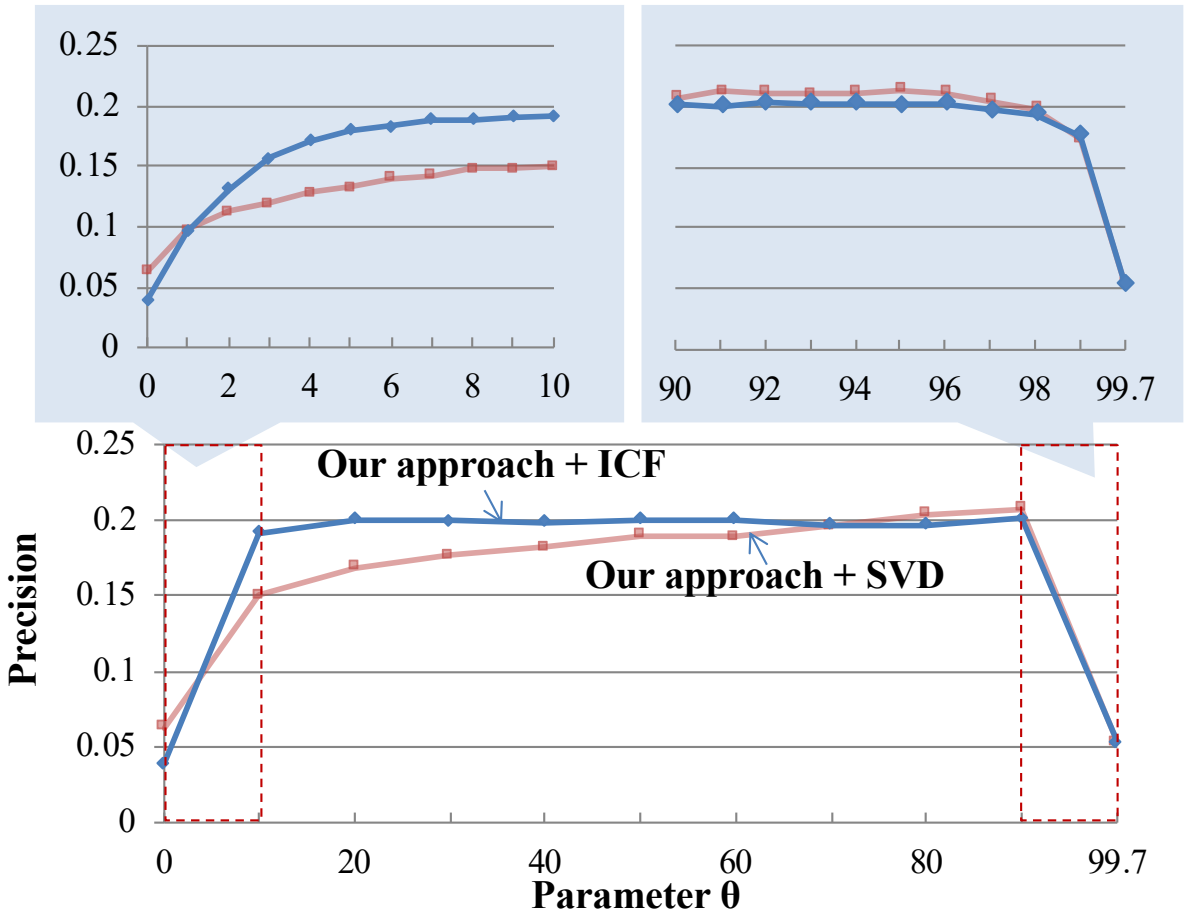


Strengths of Our Approach

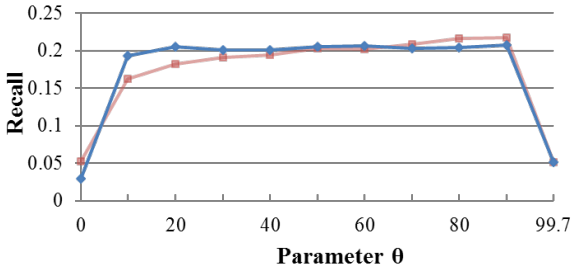
- Our approach could *improve the CF methods*
  - Uninteresting items are *excluded from a recommendation list*
  - A zero-injected matrix *provides richer information to CF*
    - With a *much more number of ratings* (including ratings with zero values)
- Our approach is *orthogonal to* existing CF methods
  - We *simply replace* the rating matrix *R* by the zero-injected matrix *Z*
  - Business can *choose any appropriate CF methods* suiting their particular settings

Effectiveness (IEEE ICDE'16 / IEEE TKDE'19)

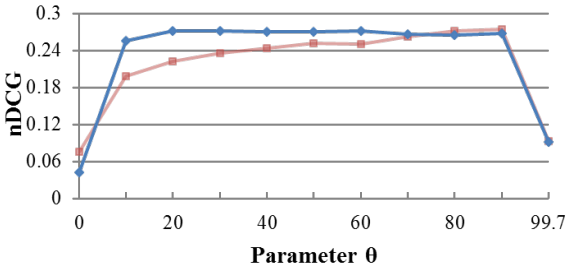
- Accuracy of ICF and SVD equipped with our zero-injection under varying parameter θ



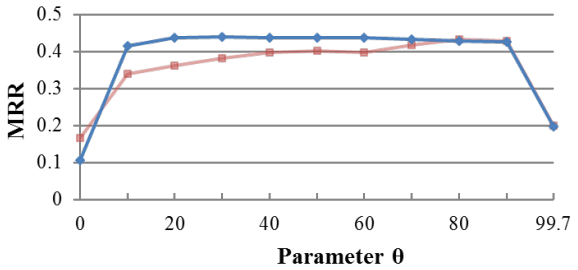
<Precision>



<Recall>



<nDCG>



<MRR>

Effectiveness (IEEE ICDE'16 / IEEE TKDE'19)

- Accuracy of four CF methods equipped with zero-injection ($\theta = 90\%$)

Metric		ICF			SVD			SVD++			PureSVD		
		Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain
Precision	@5	**0.039**	**0.201**	**413.8%**	**0.063**	**0.207**	**229.7%**	**0.076**	**0.193**	**153.3%**	**0.100**	**0.106**	**16.7%**
	@10	0.041	**0.161**	**292.6%**	0.056	**0.166**	**196.9%**	0.069	**0.154**	**123.9%**	0.082	**0.089**	**19.1%**
	@15	0.040	**0.137**	**243.7%**	0.053	**0.142**	**169.9%**	0.063	**0.134**	**112.0%**	0.071	**0.078**	**11.3%**
	@20	0.039	**0.121**	**211.7%**	0.048	**0.125**	**159.1%**	0.058	**0.118**	**102.3%**	0.063	**0.071**	**13.7%**
Recall	@5	0.030	**0.207**	**600.3%**	0.052	**0.218**	**316.0%**	0.063	**0.194**	**209.6%**	0.112	**0.120**	**16.9%**
	@10	0.059	**0.305**	**412.7%**	0.089	**0.325**	**265.9%**	0.109	**0.288**	**163.1%**	0.175	**0.191**	**19.3%**
	@15	0.085	**0.375**	**341.4%**	0.121	**0.394**	**226.3%**	0.150	**0.361**	**141.2%**	0.220	**0.245**	**11.4%**
	@20	0.111	**0.428**	**285.4%**	0.144	**0.450**	**213.5%**	0.184	**0.415**	**125.3%**	0.254	**0.293**	**15.4%**
nDCG	@5	0.043	**0.268**	**527.9%**	0.076	**0.274**	**260.7%**	0.087	**0.256**	**196.0%**	0.135	**0.143**	**16.0%**
	@10	0.053	**0.285**	**436.0%**	0.084	**0.297**	**252.0%**	0.099	**0.272**	**175.6%**	0.151	**0.162**	**17.6%**
	@15	0.062	**0.303**	**390.7%**	0.094	**0.315**	**234.8%**	0.110	**0.291**	**163.7%**	0.164	**0.178**	**18.9%**
	@20	0.071	**0.319**	**351.7%**	0.101	**0.332**	**227.3%**	0.121	**0.306**	**153.5%**	0.174	**0.193**	**10.9%**
MRR		0.106	**0.426**	**303.0%**	0.165	**0.428**	**159.2%**	0.181	**0.416**	**129.3%**	0.262	**0.274**	**14.7%**

How to Impute Missing Ratings? Claims, Solution, and Its Application to Collaborative Filtering

(WWW'18)

Data Imputation

- A solution to the data sparsity problem
  - To *infer* missing ratings from existing ratings
  - To *impute* them to rating matrix *R*
- Advantage
  - *Not to require additional information* from other sources such as trust networks and crowdsourcing

Items

Users	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1		1		1		5	3			5		
u_2	1	3		4		4				2	4	
u_3												5
u_4		3		4		4		4	3			
u_5			4									
u_6	1			3	5	1		4			3	

Rating matrix

Existing Approaches for Data Imputation

- Group M
  - To infer missing ratings with ***M**ultiple (i.e., different) values*
    - To determine the values of missing ratings *by performing CF* based on existing ratings
  - Example: AutAI [CIKM2012] and AdaM [ASONAM2013]

AutAI
AdaM

Users \ Items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	3.1	1	4.2	1	1.2	5	3	4.3	5.0	5	2.8	4.4
u_2	1	3	3.9	4	4.2	4	2.6	4.0	4.2	2	4	2.9
u_3	4.2	3.2	4.1	5.0	3.3	3.8	3.9	4	3	3.7	4.4	5
u_4	1.1	3	3.4	4	4.8	4	4.4	4.4	2.6	4.7	3.8	4.2
u_5	2.5	5.0	4	5.0	3.2	1.9	3.5	1.7	5	4.1	1.9	2.8
u_6	1	4.2	4.4	3	3	1	2.6	4	4.3	4.2	3	5.0

A Rating Matrix

Existing Approaches for Data Imputation

- Group S
  - To infer missing ratings with *a **S**ingle value*
    - To assume that missing ratings implicitly represent *a user's negative preference*
  - Example: PureSVD [RecSys2010], Zero-injection [ICDE2016], and AllRank [KDD2010]

PureSVD
Zero-injection

Users \ Items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	0	1	0	1	0	5	3	0	0	5	0	0
u_2	1	3	0	4	0	4	0	0	0	2	4	0
u_3	0	0	0	0	0	0	0	4	3	0	0	5
u_4	0	3	0	4	0	4	0	0	0	0	0	0
u_5	0	0	4	0	0	0	0	0	5	0	0	0
u_6	1	0	0	3	3	1	0	4	0	0	3	0

A Rating Matrix

Existing Approaches for Data Imputation

- Group S
  - To infer missing ratings with *a **S**ingle value*
    - To assume that missing ratings implicitly represent *a user's negative preference*
  - Example: PureSVD [RecSys2010], Zero-injection [ICDE2016], and AllRank [KDD2010]

AllRank

Users \ Items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	2	1	2	1	2	5	3	2	2	5	2	2
u_2	1	3	2	4	2	4	2	2	2	2	4	2
u_3	2	2	2	2	2	2	2	4	3	2	2	5
u_4	2	3	2	4	2	4	2	2	2	2	2	2
u_5	2	2	4	2	2	2	2	2	5	2	2	2
u_6	1	2	2	3	3	1	2	4	2	2	3	2

A Rating Matrix

Our Contributions

- To point out the limitations of existing data imputation approaches
- To suggest three new claims to overcome these limitations
- To propose a deep-learning based data imputation approach
  - Based on our hypothesis on preferences on items
  - To satisfy all of the three claims
- To perform extensive experiments with real-world datasets
  - To validate our claims and hypothesis
  - To demonstrate the effectiveness of our proposed approach

Missing Ratings: Our Observation

- Assumption
  - Users have time and budget constraints
- Characteristics of missing ratings
  - A *small number of interesting items* to each user
  - A *large number of uninteresting items* to each user

Our Three Claims

- Claim 1: *most of imputed values should be a low rating*
  - C1 is to overcome the limitation of group M (AutAI, AdaM)
  - Despite being mostly uninteresting items, group M infers most of missing ratings as high

Validation of Claim 1 (C1)

- Claim 1: *most of imputed values should be a low rating*

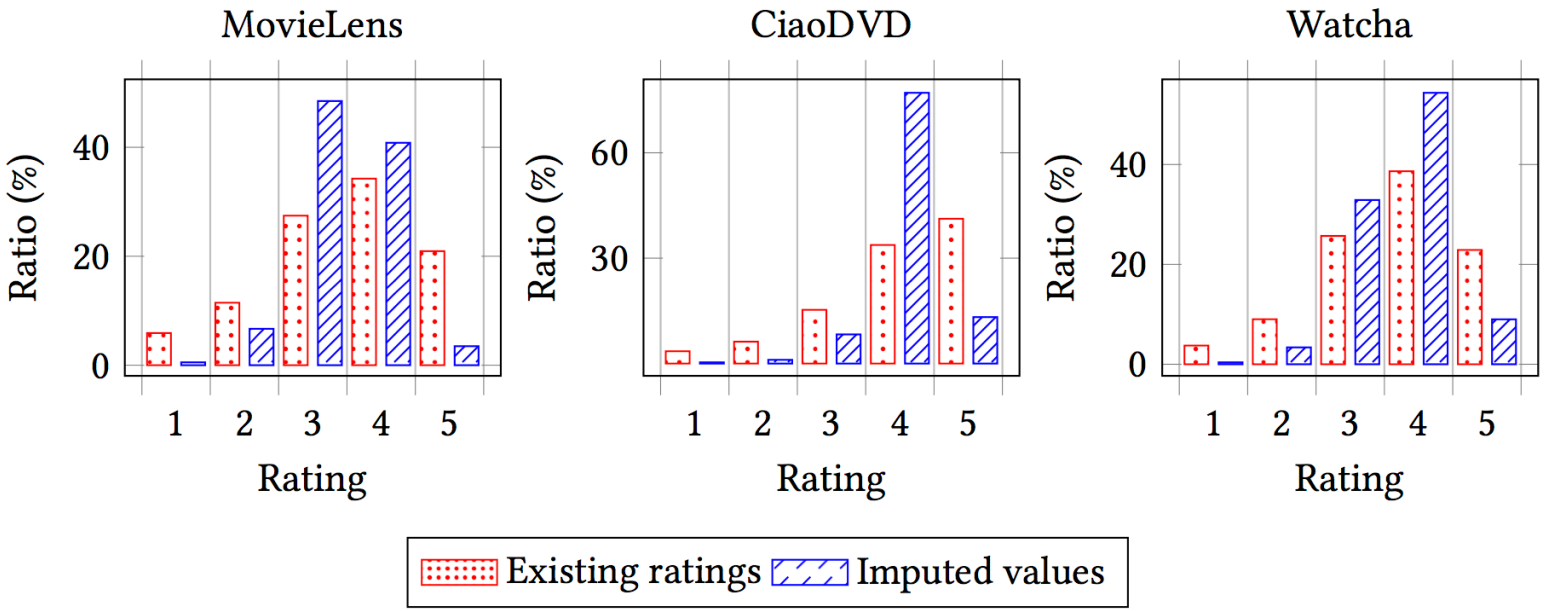


Figure 1: Distribution of existing ratings and imputed values in the three real-world datasets.

  - Imputed values are *mostly higher than or equal to 3*

Validation of Claim 1 (C1)

- Claim 1: *most of imputed values should be a low rating*

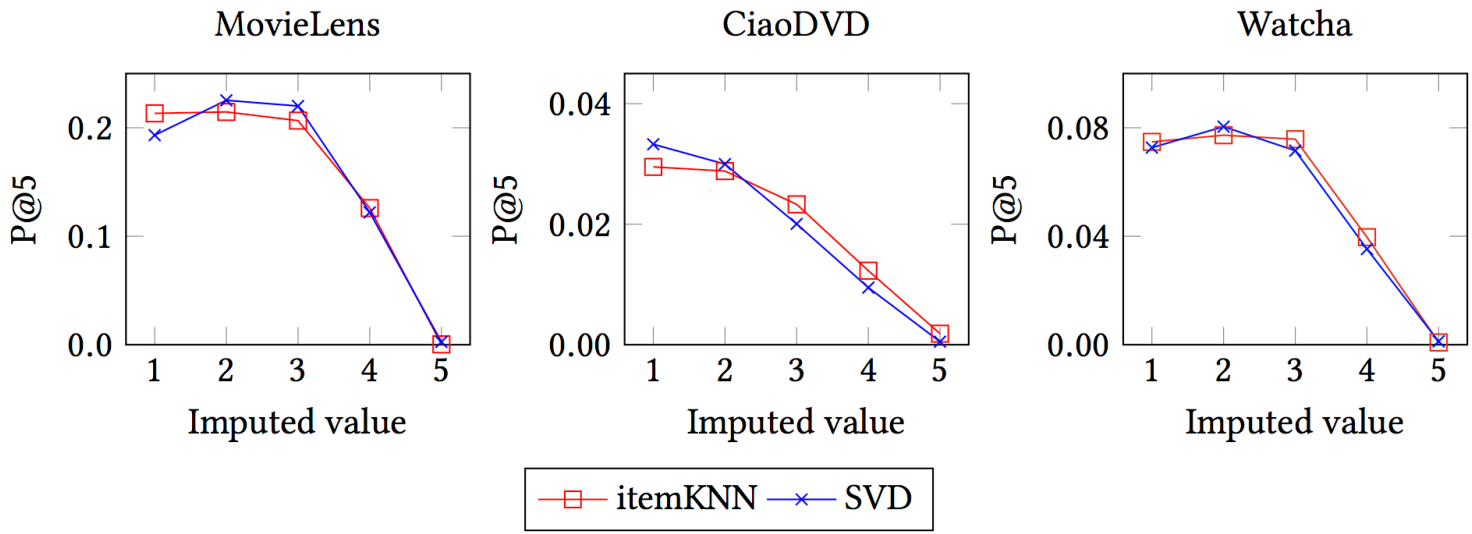


Figure 2: The accuracy with different imputed values of interesting items in the three real-world datasets.

– The accuracy of recommendations is high only when imputed values are *low ratings (i.e., 1 and 2)*

Validation of Claim 1 (C1)

- Claim 1: *most of imputed values should be a low rating*

Table 2: Comparison of group M and group S (MovieLens)

Metrics	AdaM (group M)	Zero-injection (group S)
P@5	0.051	**0.207**
P@10	0.046	**0.166**
P@20	0.042	**0.125**
R@5	0.043	**0.218**
R@10	0.076	**0.325**
R@20	0.132	**0.450**
nDCG@5	0.058	**0.274**
nDCG@10	0.066	**0.297**
nDCG@20	0.085	**0.332**

  - The accuracy of group M imputing high values is much lower than that of group S imputing low values

Our Three Claims

- Claim 1: *most of imputed values should be a low rating*
  - C1 is to overcome the limitation of group M (AutAI, AdaM)
  - Despite being mostly uninteresting items, group M infers most of missing ratings as high
- Claim 2: *the imputed values of interesting items should be higher than those of uninteresting items*
  - C2 is to overcome the first limitation of group S (Zero-injection, PureSVD, AllRank)
  - Group S ignores a small number of interesting items

Validation of Claim 2 (C2)

Hanyang Univ.

- Claim 2: *the imputed values of interesting items should be higher than those of uninteresting items*

Table 3: The accuracy with respect to different imputed values of interesting items (MovieLens)

Metrics	AllRank (I_{un} = 2)				
	I_{un}-1.0	I_{un}-0.5	I_{un}+0	I_{un}+0.5	I_{un}+1.0
$P@5$	0.081	0.128	0.184	**0.207**	**0.207**
$P@10$	0.063	0.100	0.147	**0.164**	**0.164**
$P@20$	0.047	0.075	0.108	**0.122**	**0.123**
$R@5$	0.067	0.119	0.195	**0.216**	**0.219**
$R@10$	0.099	0.174	0.287	**0.322**	**0.324**
$R@20$	0.141	0.241	0.397	**0.446**	**0.446**
$nDCG@5$	0.103	0.170	0.247	**0.278**	**0.281**
$nDCG@10$	0.104	0.175	0.264	**0.297**	**0.300**
$nDCG@20$	0.114	0.192	0.294	**0.330**	**0.334**

Metrics	pureSVD (I_{un} = 0)				
	I_{un}-1.0	I_{un}-0.5	I_{un}+0	I_{un}+0.5	I_{un}+1.0
$P@5$	0.113	0.146	0.161	**0.173**	**0.181**
$P@10$	0.087	0.115	0.128	**0.138**	**0.147**
$P@20$	0.066	0.087	0.099	**0.106**	**0.112**
$R@5$	0.102	0.142	0.163	**0.178**	**0.189**
$R@10$	0.150	0.215	0.246	**0.268**	**0.289**
$R@20$	0.212	0.302	0.360	**0.388**	**0.415**
$nDCG@5$	0.147	0.190	0.214	**0.230**	**0.242**
$nDCG@10$	0.151	0.202	0.229	**0.246**	**0.263**
$nDCG@20$	0.166	0.225	0.260	**0.280**	**0.298**

  - The accuracy increases when we *impute higher values* to interesting items than those to uninteresting items
    - Interesting items are identified by using pre-use preferences as in [ICDE2016, AAAI 2017]

Our Three Claims

- Claim 1: *most of imputed values should be a low rating*
  - C1 is to overcome the limitation of group M (AutAI, AdaM)
  - Despite being mostly uninteresting items, group M infers most of missing ratings as high
- Claim 2: *the imputed values of interesting items should be higher than those of uninteresting items*
  - C2 is to overcome the first limitation of group S (Zero-injection, PureSVD, AllRank)
  - Group S ignores a small number of interesting items
- Claim 3: *the imputed values should be within the range of ratings that users can give to items*
  - C3 is to overcome the second limitation of group S (Zero-injection, PureSVD)
  - Group S gives the rating that a user is unlikely to give to an item

Validation of Claim 3 (C3)

- Claim 3: *the imputed values should be within the range of ratings that users can give to items*

Table 4: The accuracy with respect to different imputed values ranging from 0 to 5 (MovieLens)

Metrics	Zero-injection						Metrics	pureSVD					
	0	1	2	3	4	5		0	1	2	3	4	5
P@5	0.207	0.224	**0.234**	0.224	0.159	0.031	*P*@5	0.161	0.167	0.184	**0.194**	0.105	0.005
P@10	0.166	0.175	**0.182**	0.176	0.127	0.03	*P*@10	0.128	0.136	0.147	**0.149**	0.08	0.004
P@20	0.125	0.130	**0.134**	0.131	0.096	0.03	*P*@20	0.099	0.103	0.108	**0.111**	0.06	0.003
R@5	0.218	0.235	**0.244**	0.230	0.148	0.029	*R*@5	0.163	0.176	0.195	**0.197**	0.076	0.002
R@10	0.325	0.337	**0.347**	0.330	0.224	0.058	*R*@10	0.246	0.264	0.287	**0.276**	0.113	0.003
R@20	0.450	0.464	**0.468**	0.453	0.323	0.124	*R*@20	0.360	0.379	0.397	**0.383**	0.161	0.006
nDCG@5	0.274	0.303	**0.316**	0.304	0.209	0.037	*nDCG*@5	0.214	0.224	0.247	**0.259**	0.13	0.005
nDCG@10	0.297	0.319	**0.331**	0.319	0.219	0.046	*nDCG*@10	0.229	0.242	0.264	**0.268**	0.128	0.005
nDCG@20	0.332	0.353	**0.362**	0.350	0.244	0.067	*nDCG*@20	0.260	0.274	0.294	**0.295**	0.137	0.005

  - Both Zero-injection and pureSVD show higher accuracy when *imputed values are 1, 2, and 3* rather than 0
    - This result identifies that *imputing 0 exaggerates users' negative preference* too much

Our Hypothesis for Our Approach

- *Hypothesis: most pre-use preferences on items lead to their post-use preferences*
  - Most inherent features (i.e., real content) are tightly related to external features (i.e., meta data)
  - Example: the storyline of a movie is mostly under the influence of the director, actor, and genre's characteristics

Validating Our Hypothesis

- *Hypothesis: most pre-use preferences on items lead to their post-use preferences*

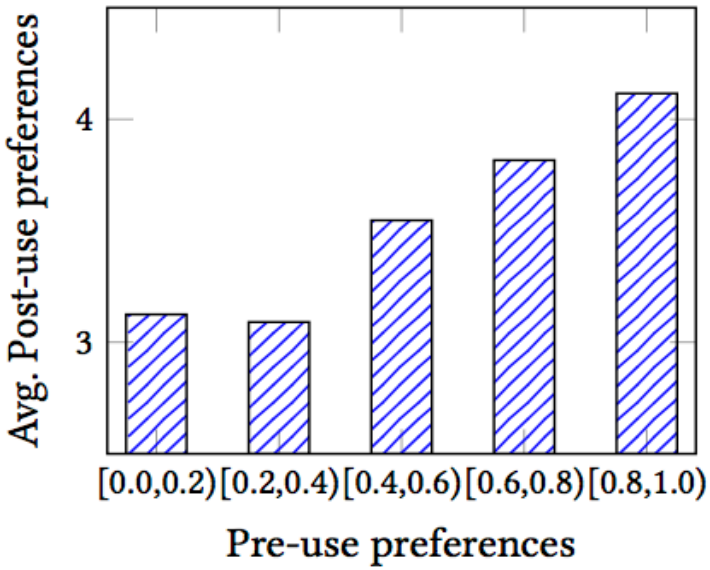


Figure 5: Correlation analysis between pre-use preferences and post-use preferences (MovieLens).

  - The positive correlation between the two axes
    - Especially, in the range of pre-use preferences between 0.2 and 1.0
    - Note that most items (around 98%) belong to the pre-use preferences range between 0.2 and 1.0

Validating Our Hypothesis

Table 5: The accuracy of top-N recommendations using pre-use preferences (MovieLens)

Metrics	itemKNN	SVD	top-*N* pre-use preferences
P@5	0.039	0.069	**0.192**
P@10	0.041	0.059	**0.151**
P@20	0.039	0.048	**0.115**
R@5	0.030	0.056	**0.202**
R@10	0.059	0.091	**0.299**
R@20	0.111	0.144	**0.421**
nDCG@5	0.043	0.084	**0.254**
nDCG@10	0.053	0.089	**0.272**
nDCG@20	0.071	0.105	**0.306**

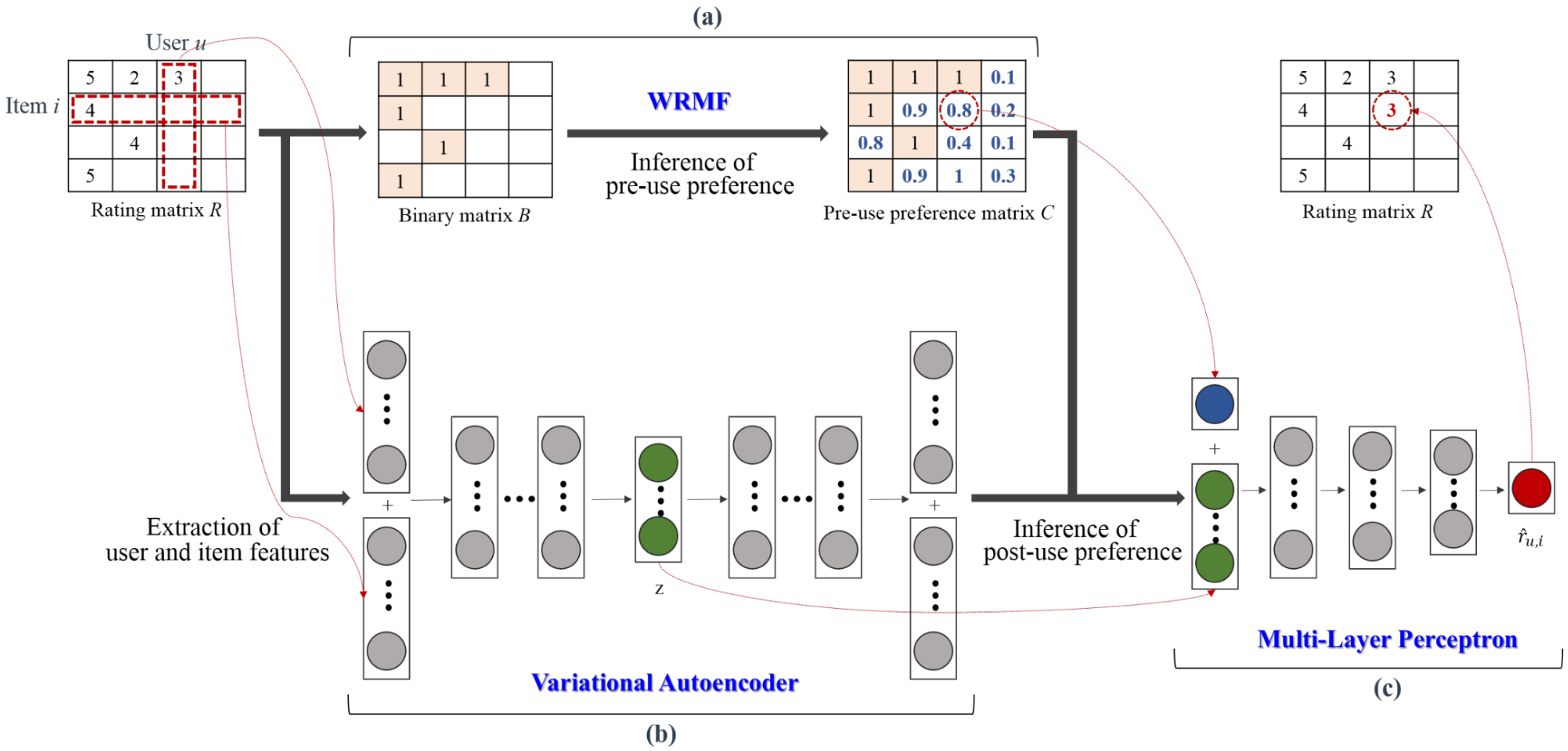
- The recommendation using pre-use preferences only (i.e., no further CF) achieves very high accuracy
  - *389% higher than itemKNN and 176% higher than SVD* (*P*@5)

Overview of Our Approach

- Deep-learning based approach
  - Based on our hypothesis
  - To reflect the *degree* of a pre-use preference of a user on an item leading to its post-use preference *for each of a user and an item*
    - The degree of a pre-use preference of a user on an item leading to its post-use preference could be different, *depending on items and users*

Our Approach

Hanyang Univ.



Strengths of Our Approach

- *CF-agnostic*
  - Applicable to any CF methods for recommendation
- To satisfy 3 claims
  - *Most imputed values are low ratings (C1)*

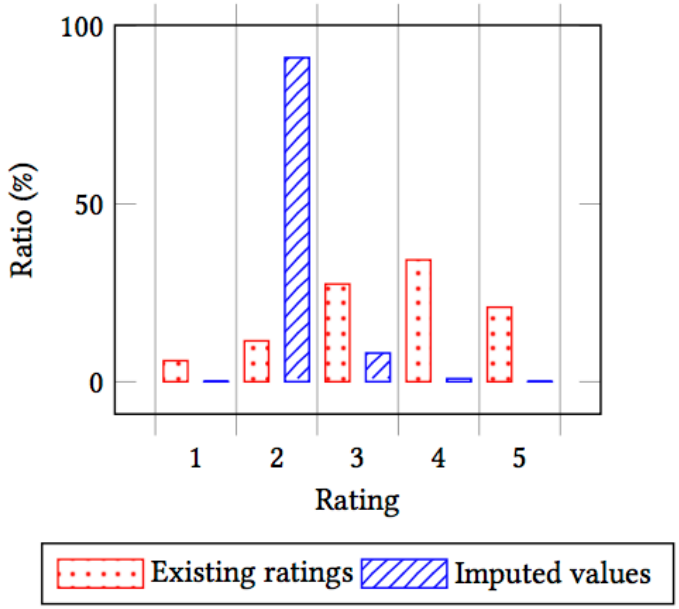


Figure 4: Distribution of imputed values inferred by our approach (MovieLens).

Strengths of Our Approach

- To satisfy 3 claims
  - *The imputed values of interesting items should be higher than those of uninteresting items (C2)*
    - Our approach infers post-use preferences by referring to pre-use preferences for pairs of a user and an item

Strengths of Our Approach

- To satisfy 3 claims
  - *Imputed values are in the range of ratings that any user actually can give to items (C3)*

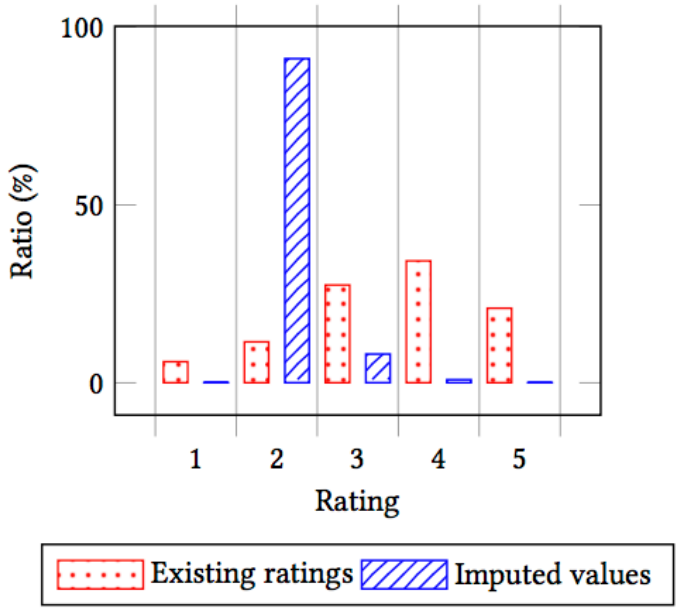


Figure 4: Distribution of imputed values inferred by our approach (MovieLens).

Comparison with State-of-the-Art Methods

- Comparison of existing approaches and our approach

Table 6: Comparison of existing state-of-the-art approaches and our approach

	Imputed values	Imputing targets
pureSVD	0	All missing ratings
AllRank	2	All missing ratings
Zero-injection	0	Ratings only for unintersting items
Our approach	[1, 5]	All missing ratings

- Two variants of our approach
  - Our approach+itemKNN, Our approach+SVD
    - To demonstrate *the effectiveness of our approach independently* of CF methods

Top-*N* Recommendation

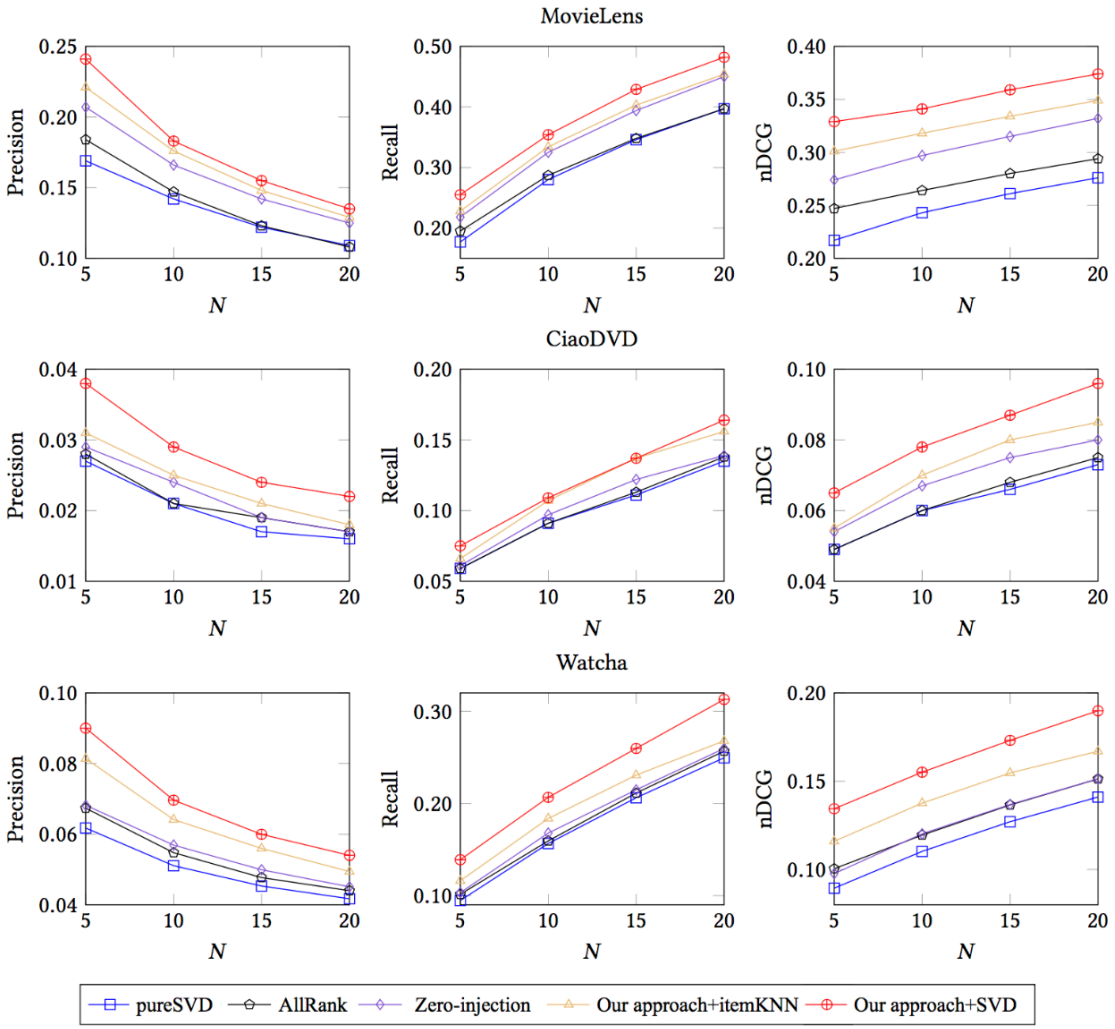


Figure 6: The accuracy of basic top-N recommendations (MovieLens, CiaoDVD, and Watcha).

Conclusions

- Summary of our contributions
  - To identify the limitations of existing data imputation approaches
  - To suggest three new claims that all data imputation approaches should follow for more accurate recommendations
  - To propose a novel deep-learning based approach, based on our hypothesis, that satisfies all of the three claims
- Results of extensive experiments with real-world datasets
  - To verify our claims and hypothesis
  - To demonstrate the superiority of our proposed approach over existing state-of-the-art approaches

gOCCF: Graph-Theoretic One-Class Collaborative Filtering Based on Uninteresting Items

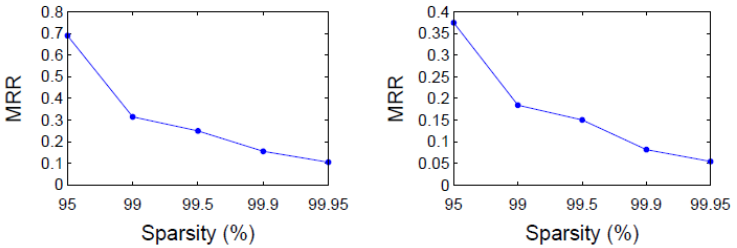
(AAAI'18)

Motivation

- ***One-Class*** Collaborative Filtering (OCCF)
  - To handle implicit feedback (*i.e.*, one-class setting, rather than ratings setting)
    - Example: Click, bookmark, and purchase
  - Challenges
    - *Less information to capture a user's taste* than in ratings setting
    - *Sparser datasets* than in ratings setting

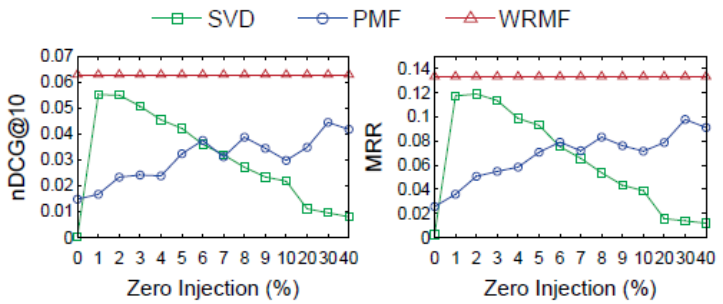
Motivation

- Existing OCCF approaches
  - Less effective in dealing with sparser datasets with many unrated items



(a) MovieLens 100K (b) Watcha

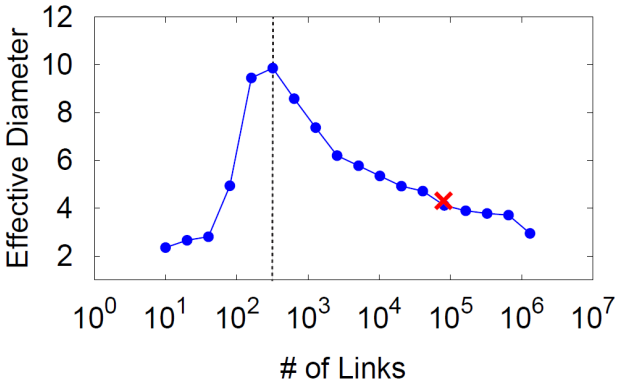
- A naïve application of zero-injection in one-class setting
  - Lower accuracy than existing OCCF approaches
  - Sensitivity to the number of uninteresting items



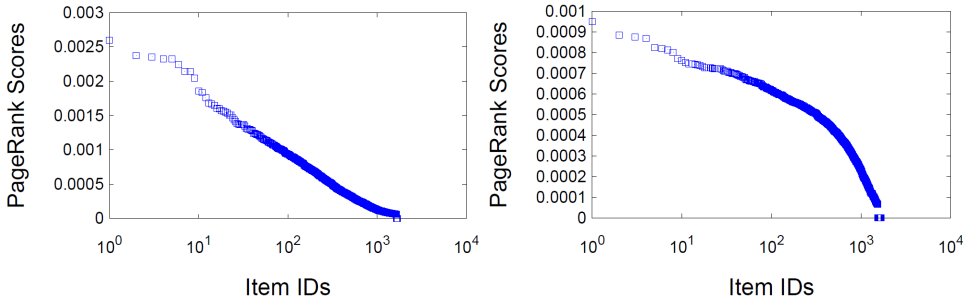
Our Idea: Exploiting Graph Properties (AAAI'18)

- Challenge: To determine *a right number of uninteresting items* with the best accuracy

- Analyze properties of negative graphs constructed by user-uninteresting item pairs
  - Topological properties
    - Use *graph shattering theory*
  - Information propagation
    - Use PageRank scores



(1) ShatterPlot

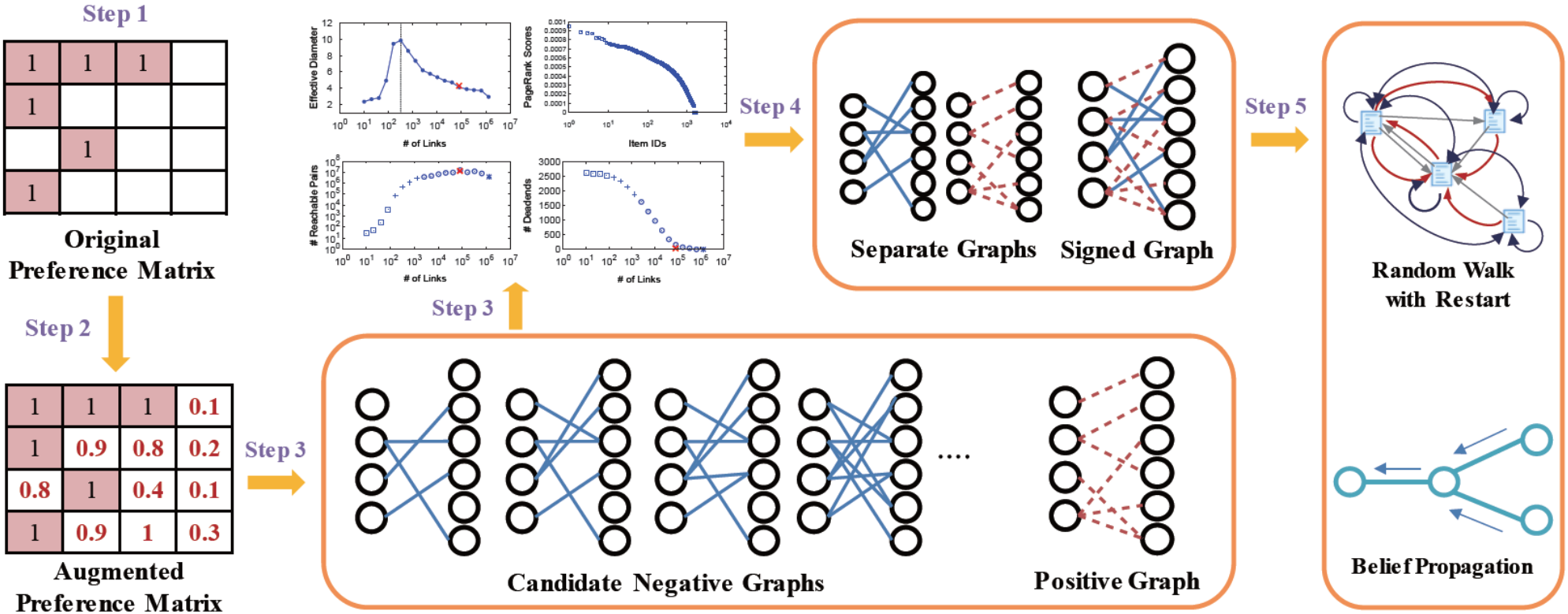


(a) Real Positive Graph (b) Negative Graph

(2) PageRank Scores

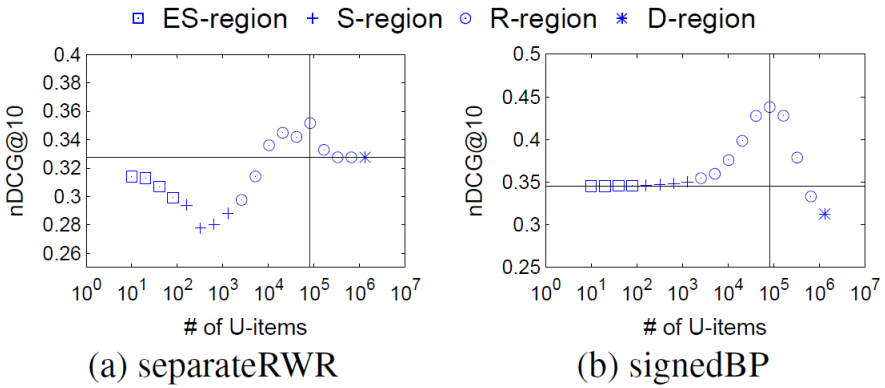
Overview of Our Approach: gOCCF (AAAI'18)

Hanyang Univ.



Effectiveness of gOCCF (AAAI'18)

- Accuracy according to the number of uninteresting items



(a) separateRWR (b) signedBP

  - Best accuracy when having the same number of negative links as that of positive links

- Accuracy before and after exploiting uninteresting items

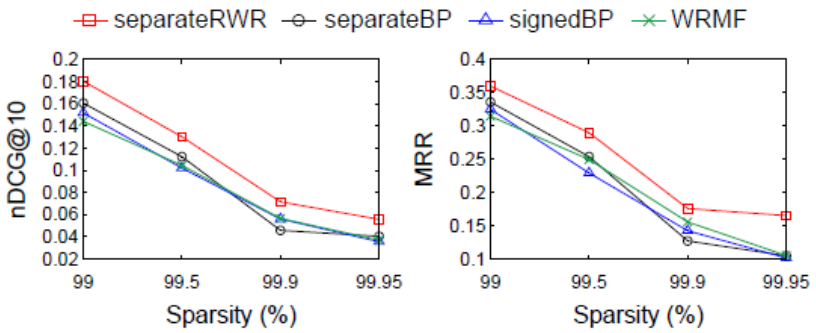
Metrics	separateRWR	separateBP	signedBP
MovieLens			
P@10	0.302 (**9.2%**)	0.309 (**7.2%**)	0.370 (**28.2%**)
R@10	0.171 (**8.2%**)	0.164 (**10.9%**)	0.210 (**42.1%**)
nDCG@10	0.352 (**7.3%**)	0.365 (**5.7%**)	0.438 (**27.0%**)
MRR	0.584 (**2.5%**)	0.604 (**1.9%**)	0.679 (**14.6%**)
HLU	43.894 (**2.8%**)	47.195 (**1.4%**)	55.980 (**20.3%**)
Watcha			
P@10	0.113 (**10.8%**)	0.124 (**13.2%**)	0.151 (**38.3%**)
R@10	0.107 (**9.6%**)	0.113 (**13.4%**)	0.142 (**41.9%**)
nDCG@10	0.136 (**9.9%**)	0.152 (**13.2%**)	0.190 (**41.8%**)
MRR	0.295 (**7.1%**)	0.329 (**10.8%**)	0.391 (**32.0%**)
HLU	14.288 (**7.7%**)	17.448 (**20.4%**)	23.012 (**58.8%**)
CiteULike			
P@10	0.122 (**21.0%**)	0.112 (**26.9%**)	0.091 (**2.9%**)
R@10	0.199 (**13.5%**)	0.162 (**26.4%**)	0.131 (**2.4%**)
nDCG@10	0.202 (**18.2%**)	0.175 (**29.9%**)	0.138 (**2.3%**)
MRR	0.326 (**14.0%**)	0.295 (**21.0%**)	0.247 (**1.3%**)
HLU	21.257 (**23.0%**)	19.323 (**33.3%**)	14.680 (**1.2%**)

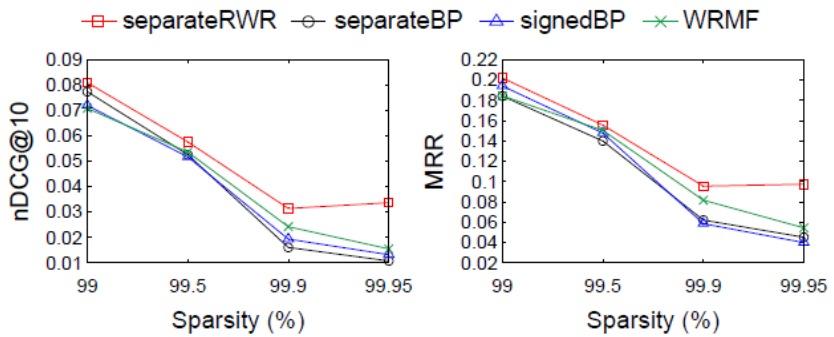
Effectiveness of gOCCF (AAAI'18)

- Accuracy of competing methods and our approach
  - CiteULike dataset

	MostPopular	SVD_ZI	PMF_ZI	WRMF	BPRMF	GBPRMF	SLIM	**separateRWR**	**separateBP**	**signedBP**
P@10	0.012	0.043	0.034	0.045	0.092	0.049	-	**0.122**	0.112	0.091
R@10	0.029	0.044	0.037	0.049	0.140	0.078	-	**0.199**	0.162	0.131
nDCG@10	0.023	0.055	0.044	0.062	0.136	0.066	-	**0.202**	0.175	0.138
MRR	0.050	0.117	0.073	0.133	0.240	0.132	-	**0.326**	0.295	0.247
HLU	1.527	5.899	4.489	7.198	12.754	4.005	-	**21.257**	19.323	14.680

- Accuracy of WRMF and our approach per sparsity
  - MovieLens 100K dataset
  - Watcha dataset





Conclusions

- We propose a *new concept of uninteresting items* to make more accurate CF for both *ratings and one-class settings*
- Our approach
  - To identify uninteresting items
  - To apply them to existing CF methods
  - To exclude them from the final recommendation list
- Strengths of our approach
  - *Orthogonal to* CF methods
  - *Parameter-free* for both *ratings and one-class settings*
  - *Consistently and universally improves* existing CF and OCCF methods

Contents

- Recommendation Systems: An Overview
  - Concepts and applications
  - Collaborative filtering (CF)
- Exploiting Uninteresting Items for Effective CF
  - Ratings setting
  - One-class setting
- **Some More Research Results**
  - Recommendations
  - Graph engines
- Conclusions

No, That's Not My Feedback: TV Show Recommendation Using Watchable Interval

(IEEE ICDE'19)

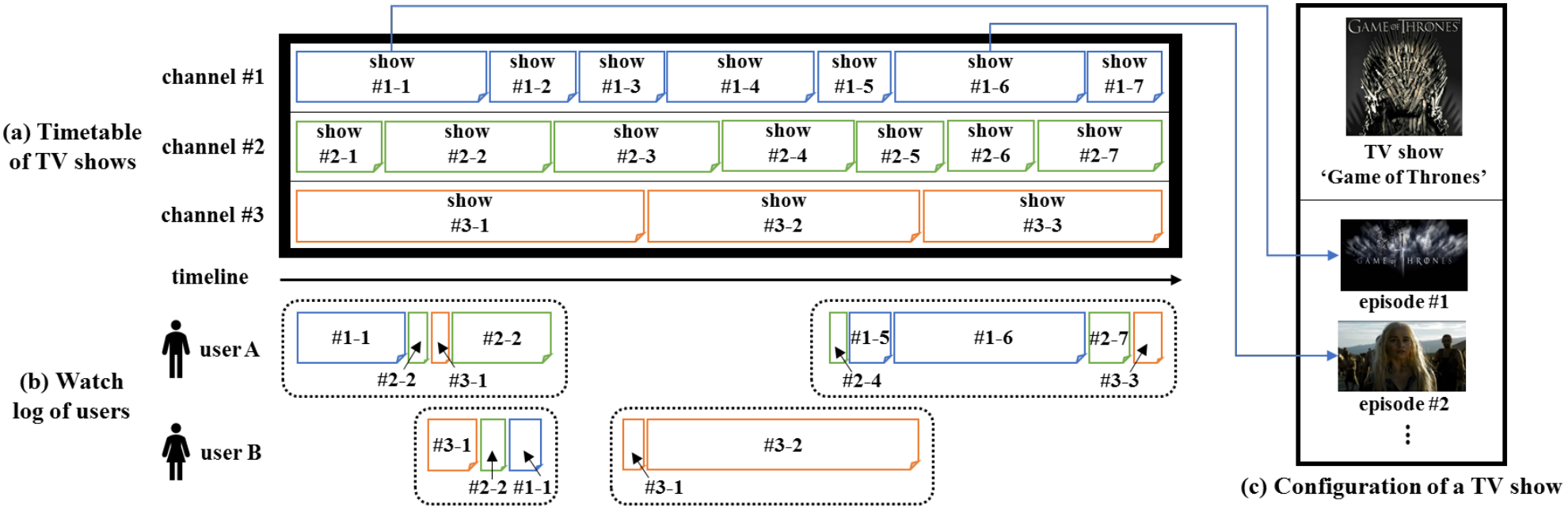
Motivation

- A number of channels (more than 100 channels)
  - A user needs to select the TV shows that she prefers
    - e.g., changes and selects channels
  - *Time-consuming* for a user to find the TV shows which she is interested in

- Requirements of a TV show recommendation system
  - Consideration of user preferences
  - Consideration of currently broadcasting shows

Characteristics of TV Show Data

Hanyang Univ.



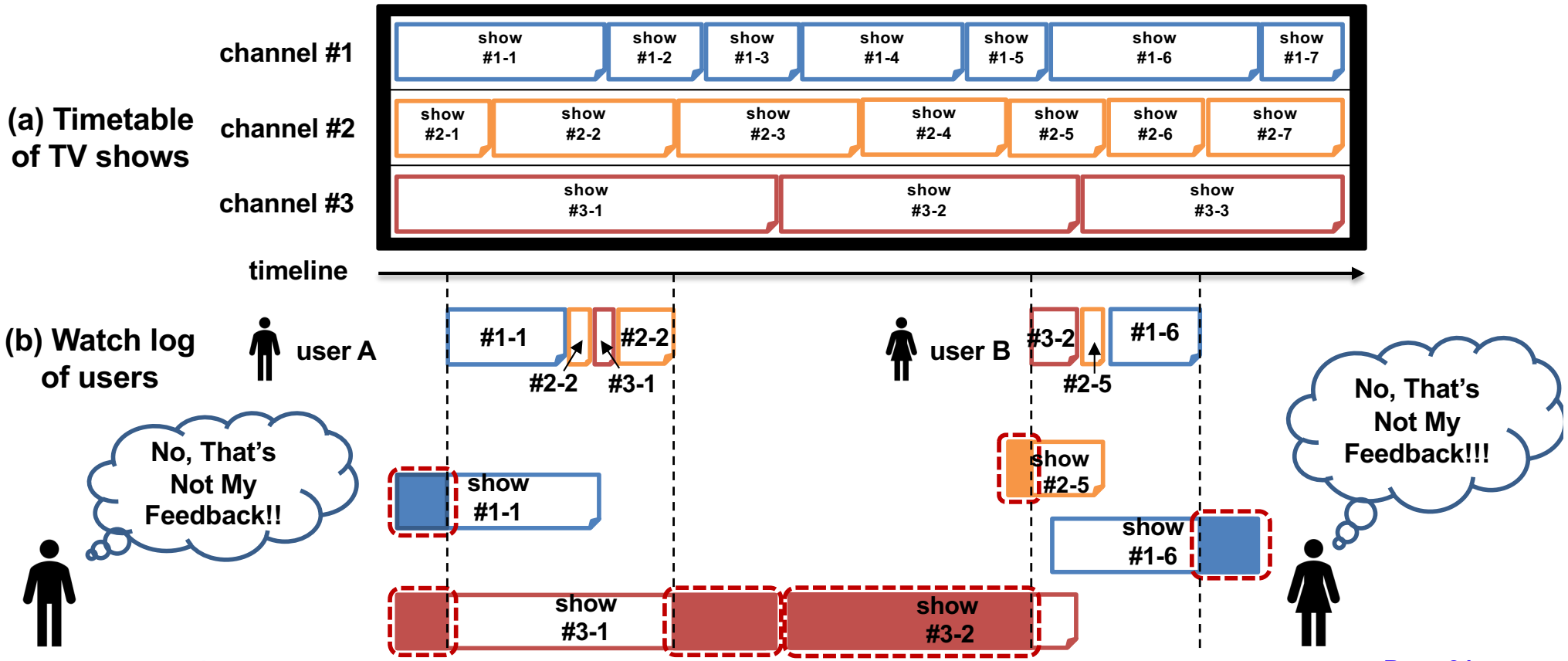
- User
  - Different watching intervals (of users) for TV
  - Different watching intervals for episodes of TV shows
    - Start at different times
    - Have watching times of different lengths

Existing Work

- Different approaches to infer a user's preference on a TV show
  - PM (Positive, Missing) [ICDM'08] - WRMF
    - Consider whether a user *watched a TV show or not* (i.e., one-class setting)
  - ShowTime [Info. Sci.'14]
    - Consider the sum of a user's *watching times for a TV show*
  - PNM (Positive, Negative, Missing) [RecSys'11] - PMF
    - Consider the ratio of a user's *watching times to the broadcasting times of a TV show*
  - RecTime [Info. Sci.'17] – PARAFAC
    - Consider the ratio of a user's *watching times to the broadcasting times of a TV show*
    - Consider the entrance and leaving time of the user for the show

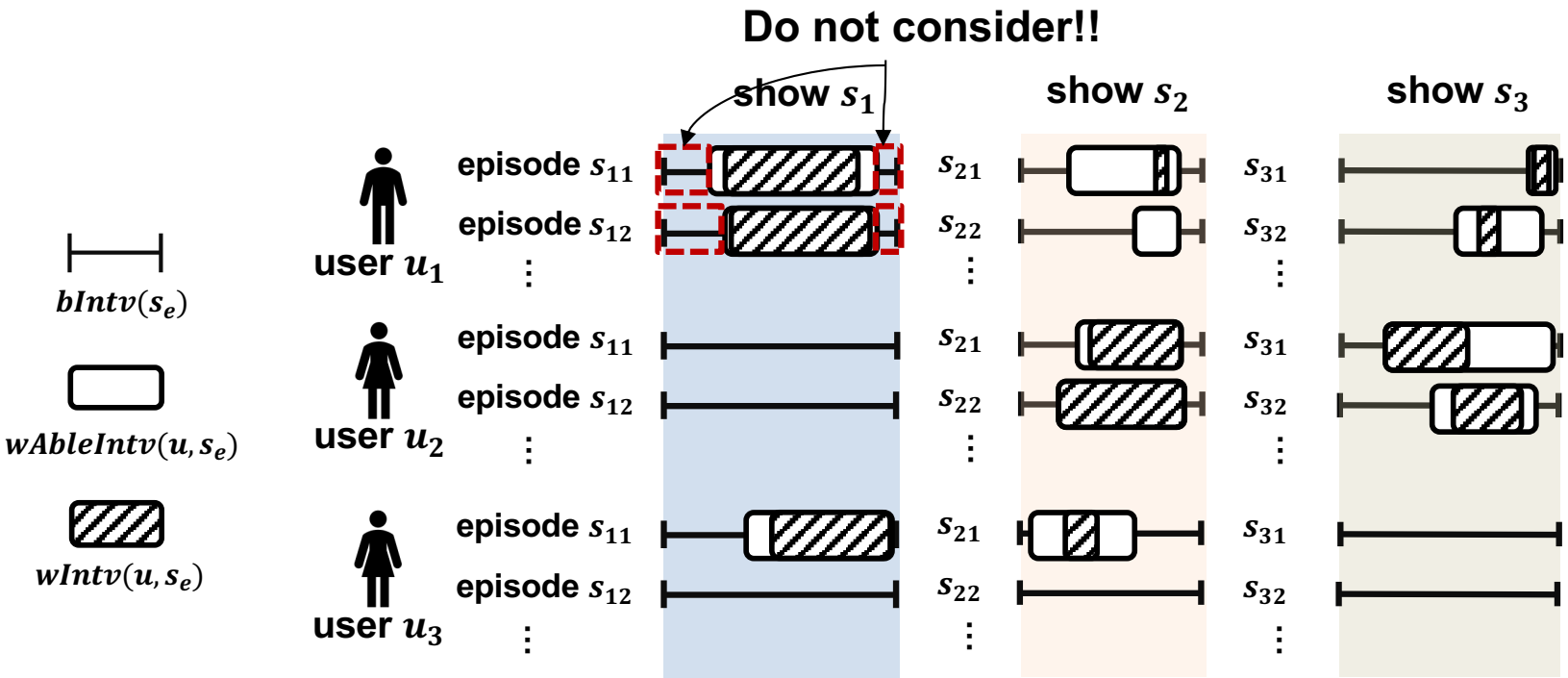
Limitation of Existing Work

- Exploit the feedback that is not expressed by users
  - A part of broadcasting intervals for the show's episodes that a user was not able to watch



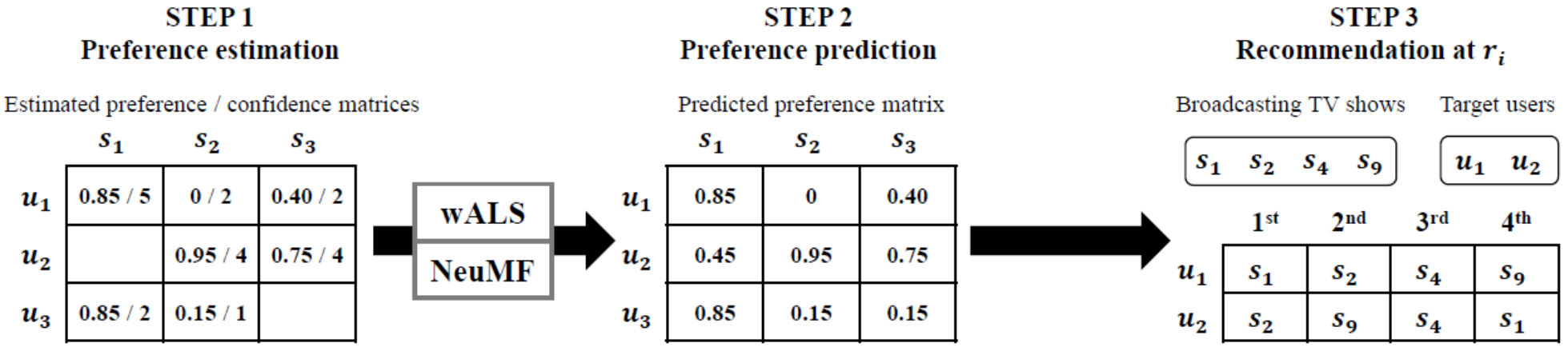
Our Idea: Considering Watchable Interval (ICDE'19)

- Consider how long a user watched an episode of a TV show during her "*watchable interval*" on it
  - The overlapped interval between her TV watching interval and the broadcasting interval of the episode



Overview of Proposed Framework (ICDE'19)

Hanyang Univ.



- Step 1: Preference estimation for watchable TV shows
- Step 2: Preference prediction for non-watchable TV shows
- Step 3: Recommendation at recommendation time

Effectiveness of Our Framework (ICDE'19)

- Estimation accuracy for watchable TV shows

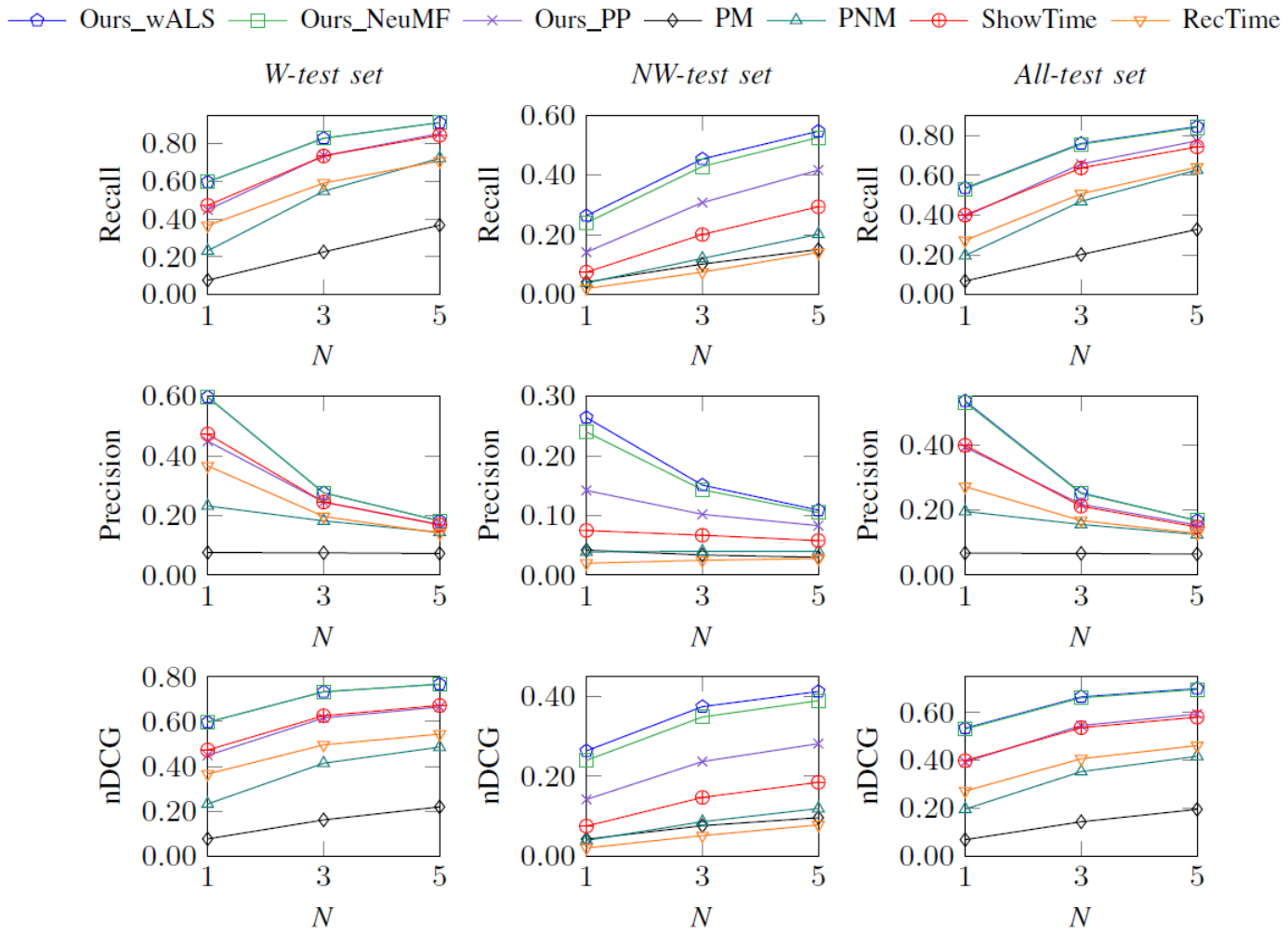
Metrics	PE	PM	PNM	ShowTime	RecTime	Popular
Recall@1	**0.449**	0.076	0.232	0.305	0.330	0.047
Recall@5	**0.854**	0.368	0.723	0.740	0.755	0.363
Precision@1	**0.449**	0.076	0.232	0.305	0.330	0.047
Precision@5	**0.170**	0.073	0.144	0.148	0.151	0.072
nDCG@1	**0.449**	0.076	0.232	0.305	0.330	0.047
nDCG@5	**0.665**	0.219	0.486	0.530	0.550	0.204
MRR	**0.620**	0.233	0.441	0.492	0.512	0.214

- Prediction accuracy for non-watchable TV shows

Metrics	PP	PM	PNM	ShowTime	RecTime	Popular
Recall@1	**0.142**	0.042	0.040	0.018	0.020	0.005
Recall@5	**0.417**	0.151	0.203	0.138	0.141	0.096
Precision@1	**0.142**	0.042	0.040	0.018	0.020	0.005
Precision@5	**0.083**	0.030	0.040	0.027	0.028	0.019
nDCG@1	**0.142**	0.042	0.040	0.018	0.020	0.005
nDCG@5	**0.282**	0.096	0.120	0.076	0.078	0.049
MRR	**0.278**	0.132	0.149	0.113	0.115	0.094

Effectiveness of Our Framework (ICDE'19)

- Accuracy of the state-of-the-art methods and our frameworks

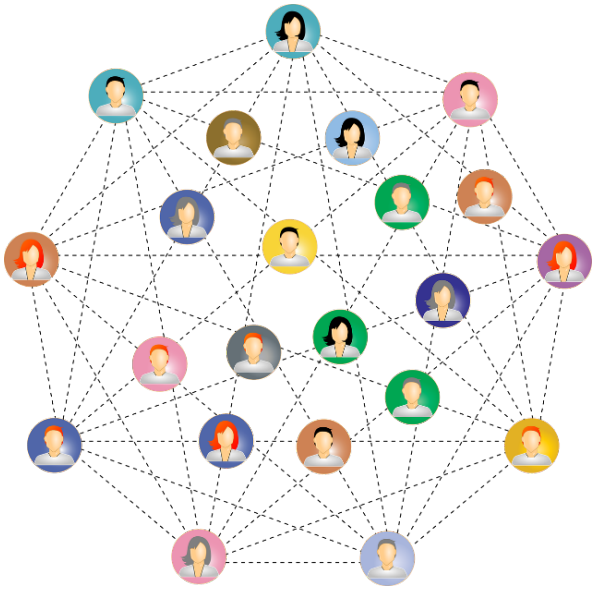


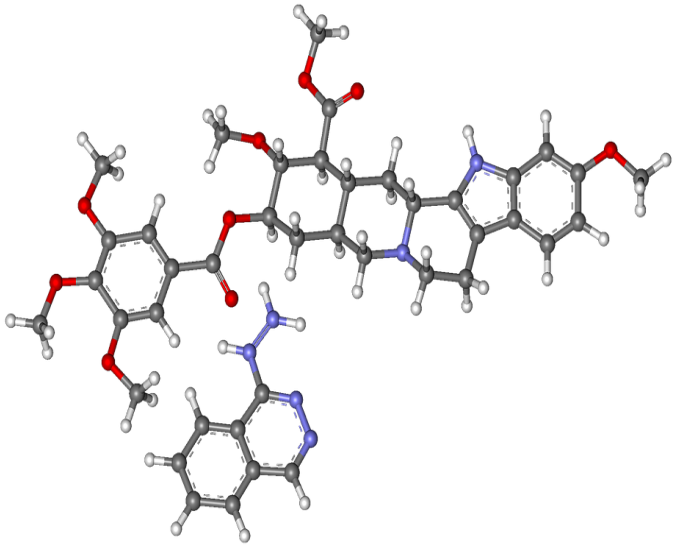
RealGraph: A Graph Engine Leveraging the Power-Law Distribution of Real-World Graphs

(WWW'19)

Graph

- Data structure to represent social networks composed of objects and their relationships
- Ubiquitous big data in real-world domains
  - Web, social networks, protein, recommendation

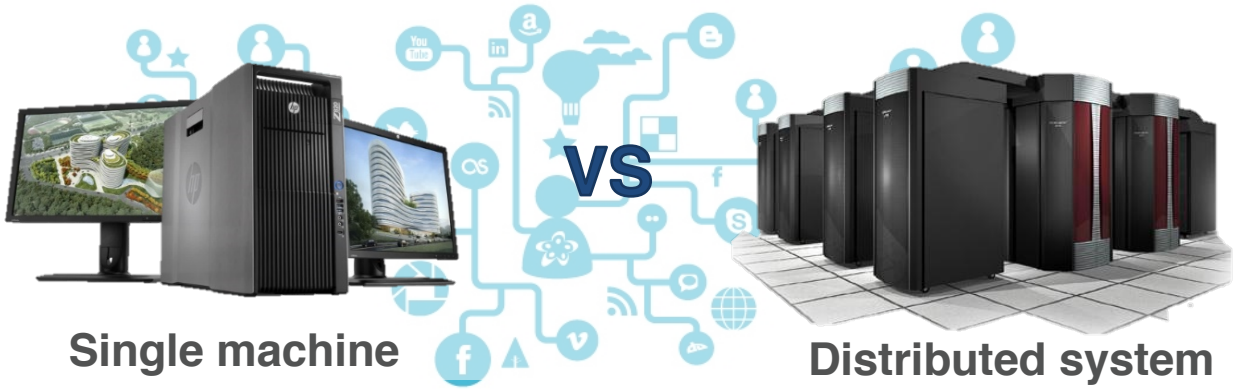




https://hackernoon.com/social-network-development-types-challenges-technologies-costs-1e185da3b2a9

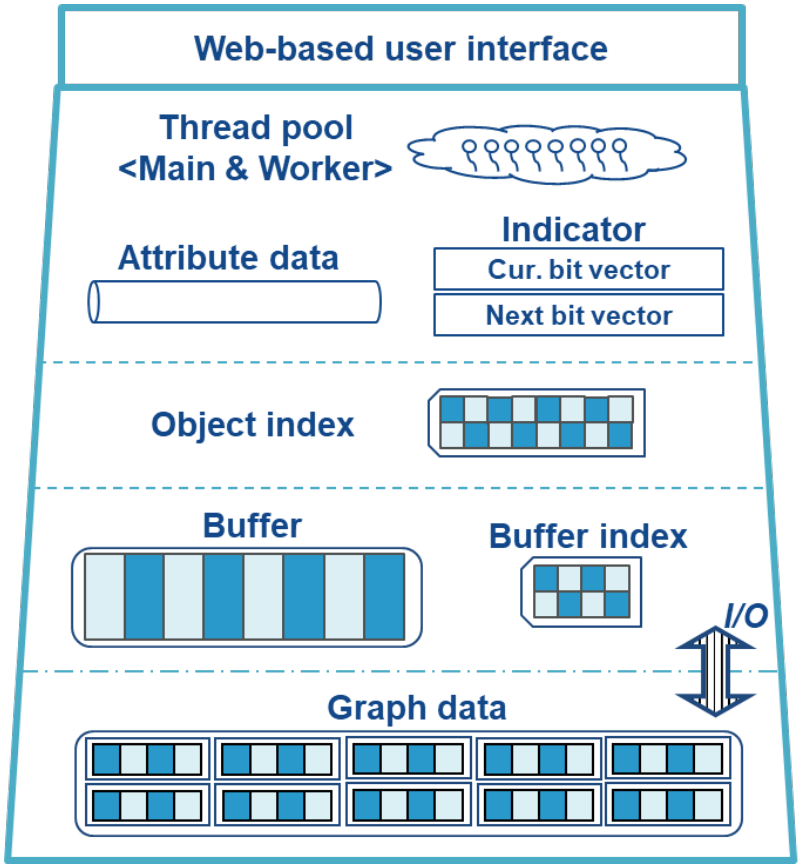
Graph Engine

- System software for graphs
  - To help programmers to handle graphs conveniently
  - To process big graphs efficiently
- Single-machine based graph engines
  - Enough computing resources to store big graphs
  - No communication overhead



RealGraph

- Single-machine based graph engine to efficiently handle big real-world graphs
  - Lower three layers
    - Storage, buffer, and object management layers
    - In charge of managing memory and storage space
    - Based on the philosophy of database systems
  - Thread management layer
    - In charge of managing threads to execute graph algorithms
  - Web-based user interface



Motivation

- No careful consideration of the power-law degree distribution of real-world graphs
  - A small number of hub nodes and a large number of non-hub nodes
- Three problems
  - Poor load balancing; inefficient node search; inefficient data layout

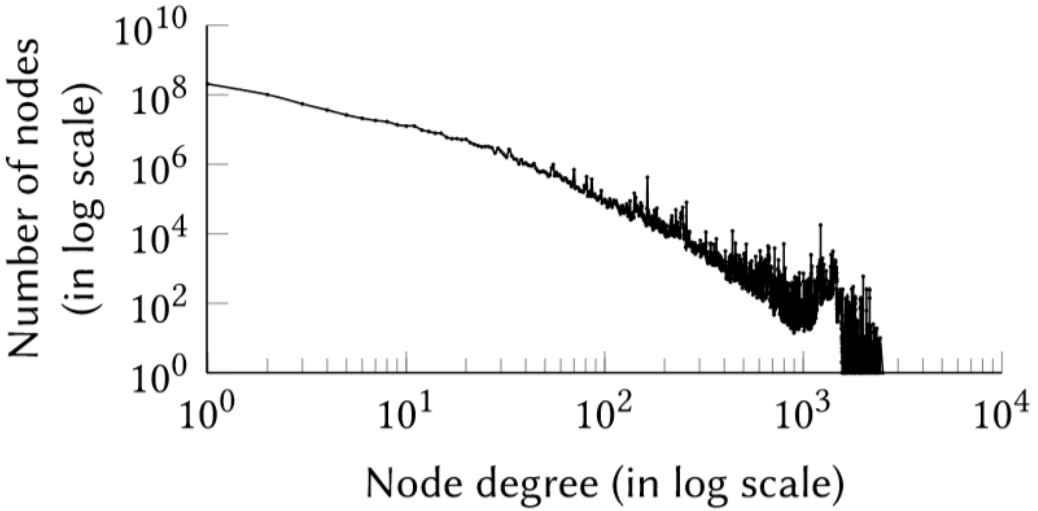
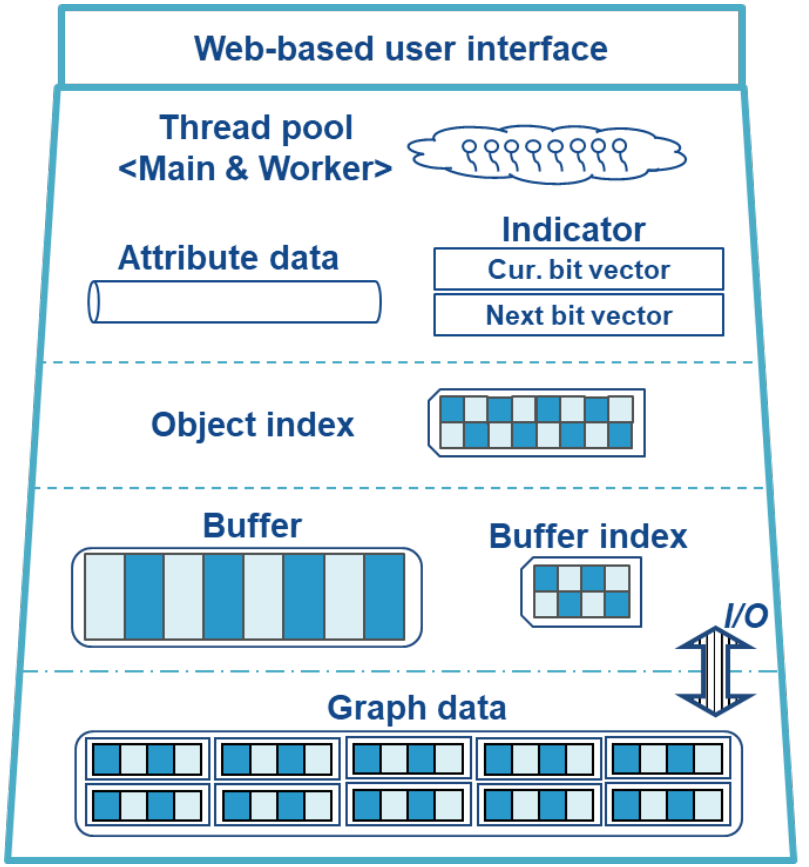


Figure: Degree distribution of the Yahoo dataset.

- Single-machine based graph engine to efficiently handle big real-world graphs
  - Lower three layers
    - Storage, buffer, and object management layers
    - In charge of managing memory and storage space
    - Based on the philosophy of database systems
  - Thread management layer
    - In charge of managing threads to execute graph algorithms
  - Web-based user interface



Evaluation

- Comparison with existing distributed-system based graph engines
  - RealGraph provides better performance in most cases
  - With much smaller computing resources

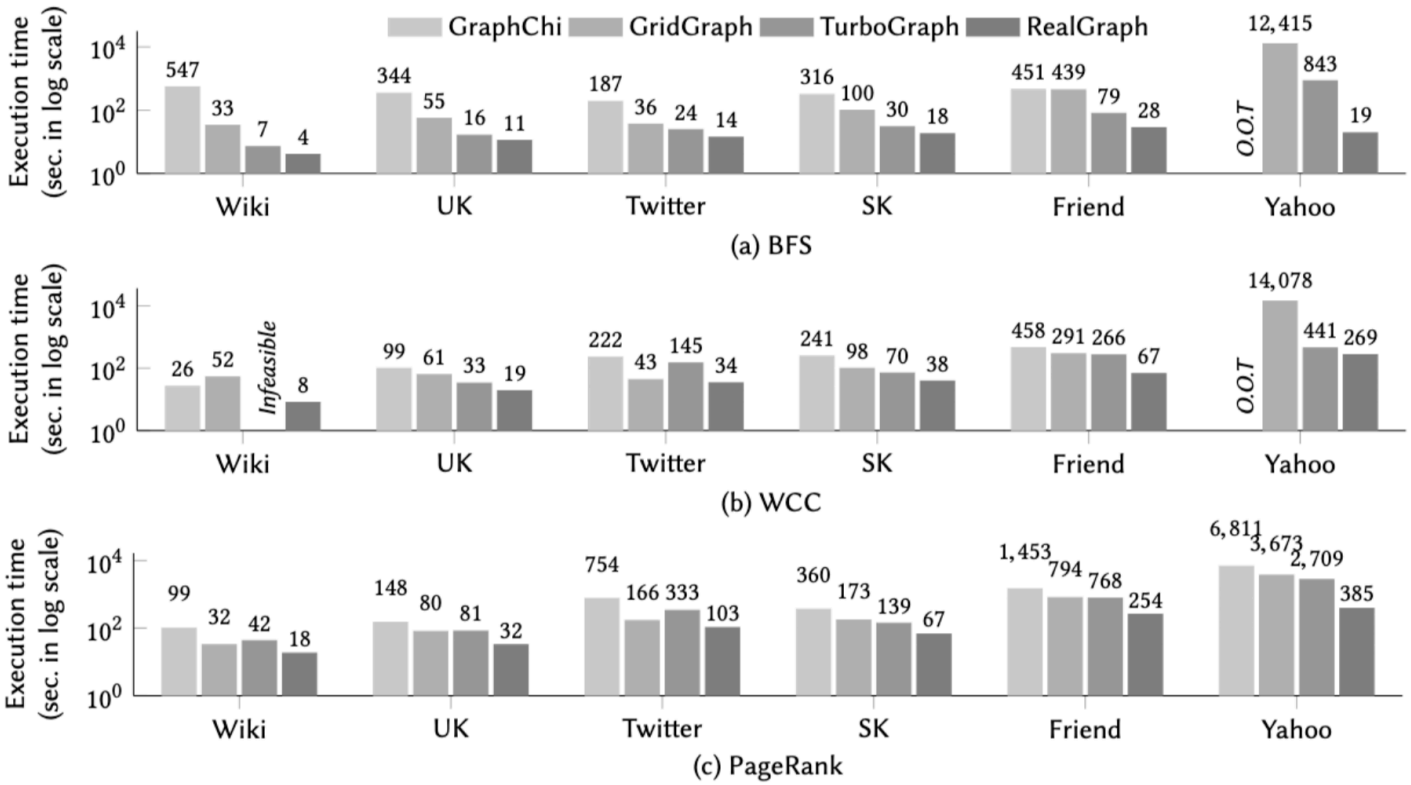
Quoted results on common algorithms on the Twitter dataset (unit: seconds)

	Graph Algorithms		System Environment		
Graph Engines	WCC	Page Rank	Machine	#Cores	Memory Size
RealGraph	34	94	Single PC with Intel i7	4	16G
PowerGraph	-	72	64 Amazon EC2 cc1.4xlarge	8	23G
GraphLab	244	249	16 Amazon EC2 cc2.4xlarge	8	68G
GraphX	251	419	16 Amazon EC2 cc2.4xlarge	8	68G
Giraph	200	596	16 Amazon EC2 cc2.4xlarge	8	68G

hyphen (-) means that the execution time is not measured

Evaluation

- Comparison with existing single-machine based graph engines
  - RealGraph provides the best performance universally
  - RealGraph outperforms the best of existing ones by 1.8 ~ 44 times



Conclusions

- Recommendation Systems: An Overview
  - Concepts and applications
  - Collaborative filtering (CF)
- Exploiting Uninteresting Items for Effective CF
  - Ratings setting
  - One-class setting
- Some More Research Results
  - Recommendations
  - Graph engines

Data Science Lab. @ Hanyang University

Hanyang Univ.

- Since Mar. 2003

Thank You !