Introduction to Recommender System

Presentor: Jinduk park

Contents

Part1. Introduction to Recommender Systems

1st session : Introduction to RS

- Collaborative filtering vs Content-based filtering
- Deep learning based recommender system
- Session-based recommender systems
- Cross-domain recommender systems

2nd session : SVD++, a powerful RS model

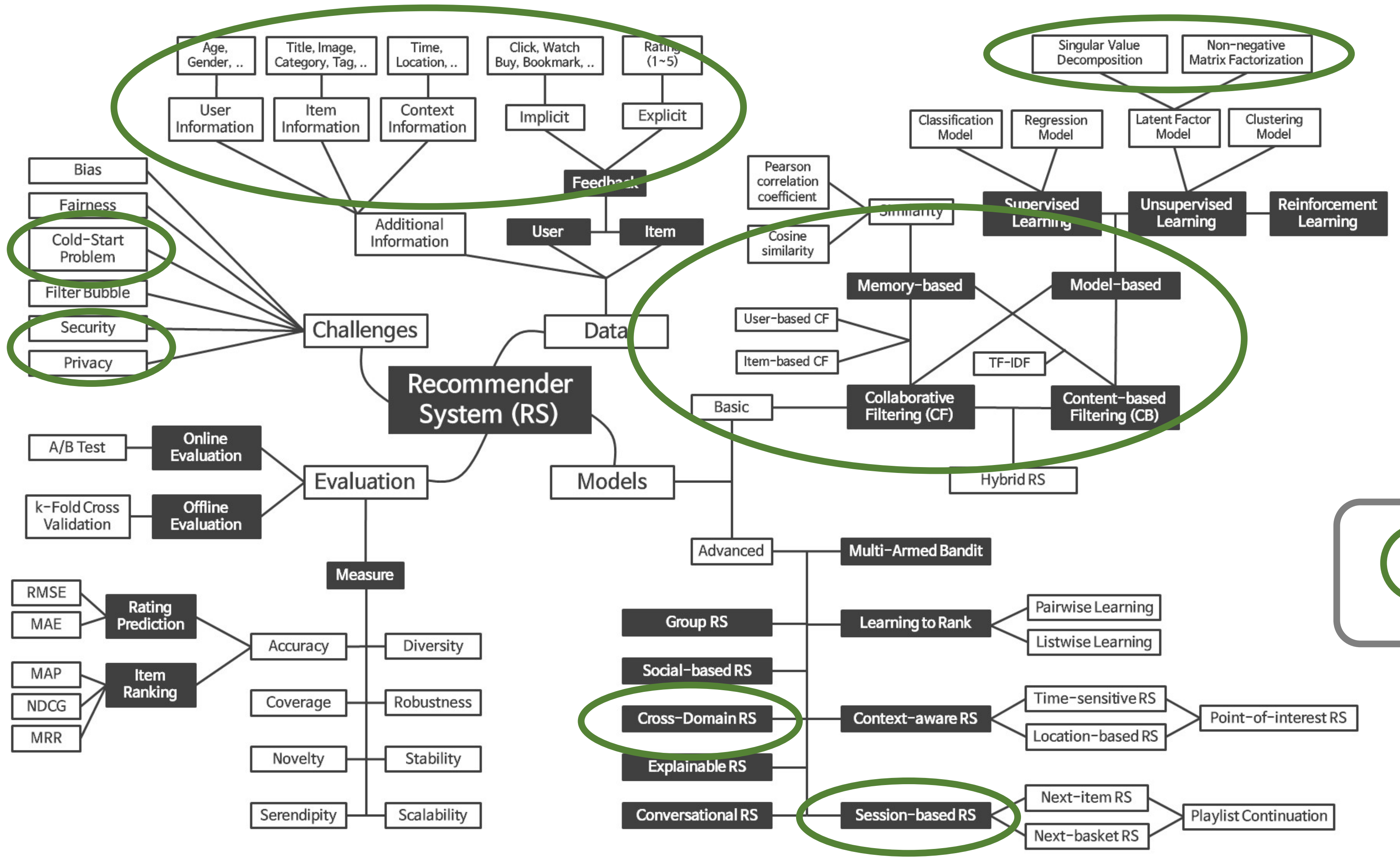
- Background
- Motivation
- Methodology

Part2. Graph Neural Networks in Recommender Systems
(will be covered next time)

- TBD

1st session : Introduction to RS

Taxonomy of RS



https://github.com/jihoo-kim/awesome-RecSys#1-books

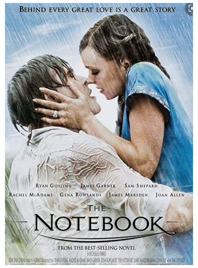
There are **broad fields** of RS, according to different perspectives.

: Contents that will be covered in this presentation

Collaborative filtering

User 1

User 2



User 3

User 4

Collaborative filtering

User 1

User 1 and 3 have similar movie taste!

User 3

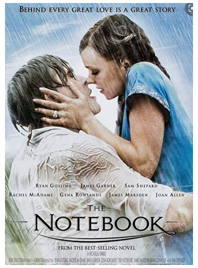
Recommends
<Betman>

This is **user-based collaborative filtering** (CF)

Collaborative filtering

User 1

User 2

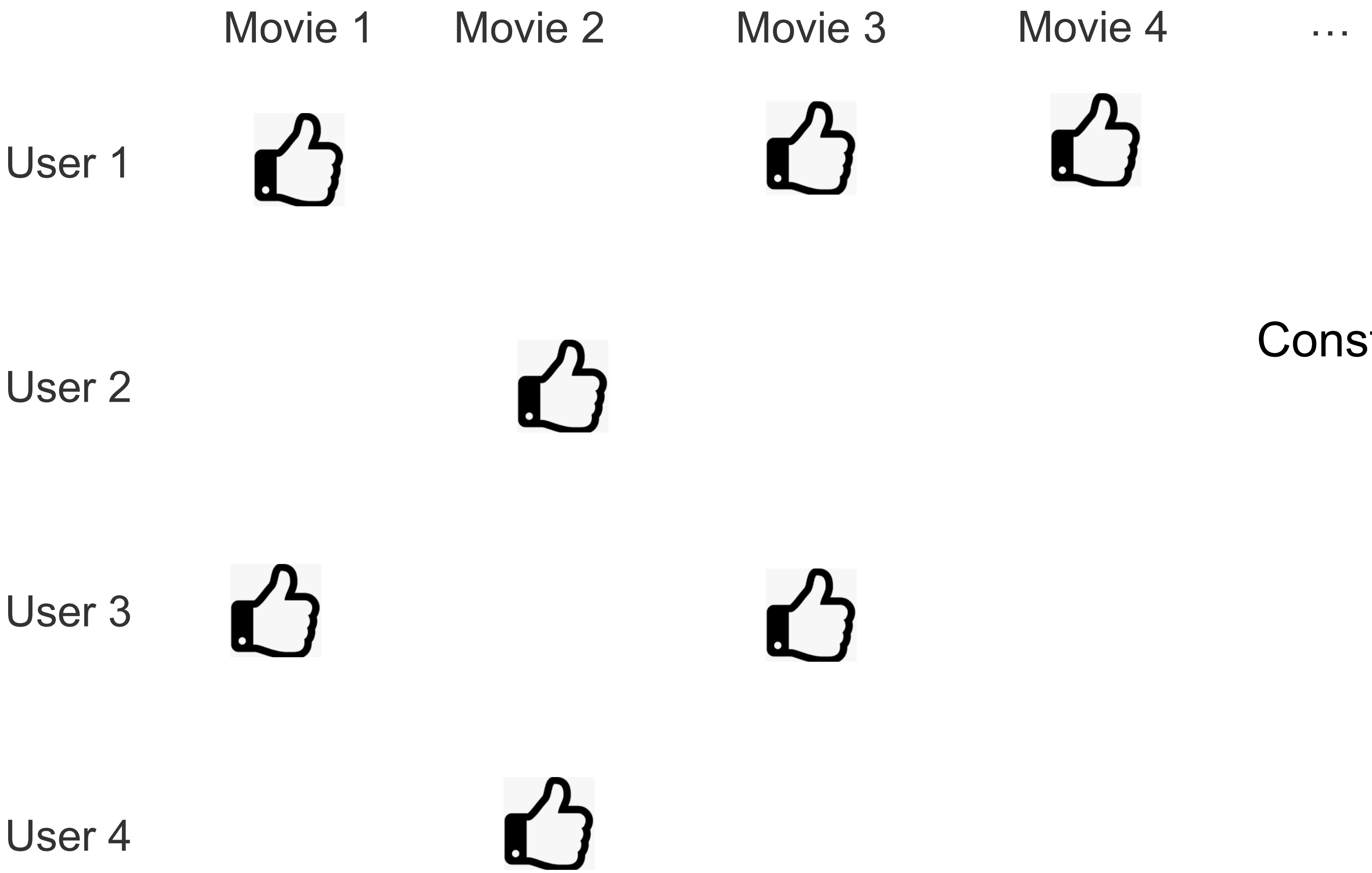


User 3

Then, how can we find *similar* user, explicitly?

User 4

Collaborative filtering



Construct **user-item matrix** first

Collaborative filtering

	Movie 1	Movie 2	Movie 3	Movie 4	…
User 1	1	0	1	1	
User 2	0	1	0	0	
User 3	1	0	1	0	
User 4	0	1	0	0	

Encode the feedback into values.

Collaborative filtering

	Movie 1	Movie 2	Movie 3	Movie 4	…
User 1	1	0	1	recommended **1**	
User 2	0	1	0	0	
User 3	1	0	1	0	
User 4	0	1	0	0	

Define and calculate similarity of user vector

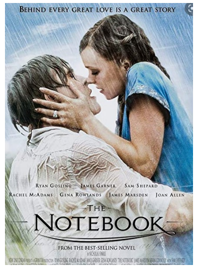
Ex) cosine similarity of user1 and user3

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Collaborative filtering

User 1

User 2

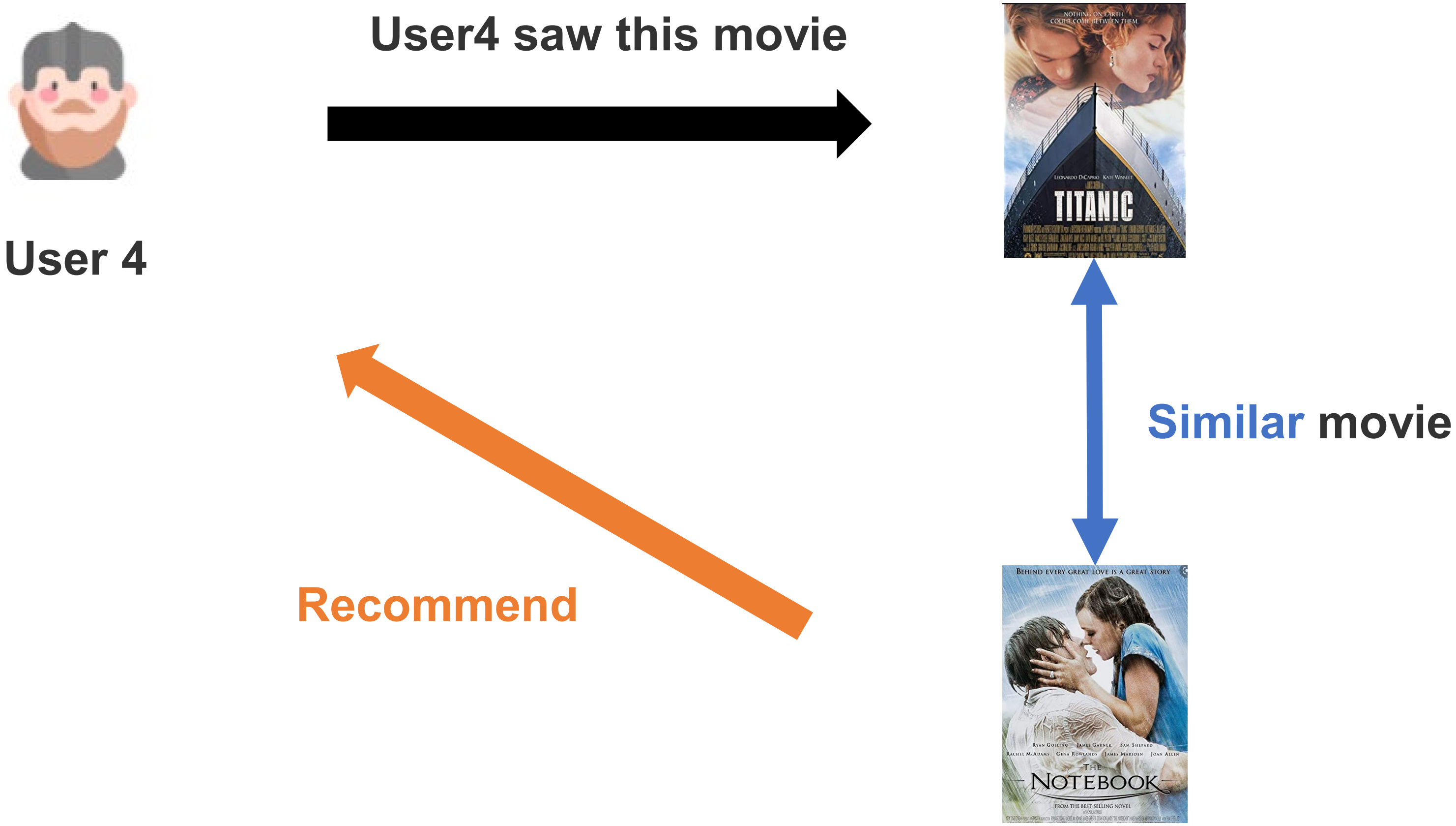


User 3

What movie would you recommend for user 4?

User 4

Collaborative filtering



This is **item-based collaborative filtering**

Collaborative filtering

	User 1	User 2	User 3	User 4	…
Movie 1	1	0	1	0	
Movie 2	0	1	0	0	
					…
Movie 3	1	0	1	0	
Movie 4	1	0	1	1	

How can we explicitly find *similar* item in item-based CF?

Let's switch row & column of the matrix.

Collaborative filtering

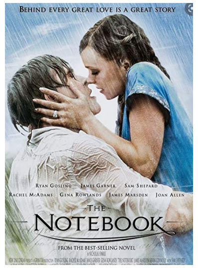
	User 1	User 2	User 3	User 4	…
Movie 1	1	0	1	0	
Movie 2	0	1	0	0	…
Movie 3	1	0	0	0	
Movie 4	1	0	1	1	

e.g.)
If cosine similarity of Movie 1 and Movie 4 is the highest
-> recommend Movie 1 !

Cold-start problem

User 1


User 2

What if new item / user comes?
(cold-start problem)

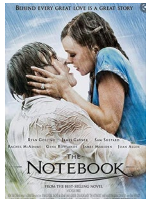
User 3


User 4


New movie release

Cold-start problem

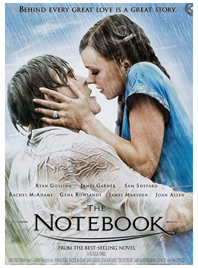
Utilize **meta-info of each item** as features, and find similar item!

	Action	Romance	Comedy	Real story-based
	1	0	0	0
	0	1	1	0
	1	0	0	0
	0	1	1	0

Cold-start problem

User 1

User 2



This is ***content based filtering*** and
It can be a solution for cold-start problem.

User 3

Recommend

User 4

New movie release

Session-based RS

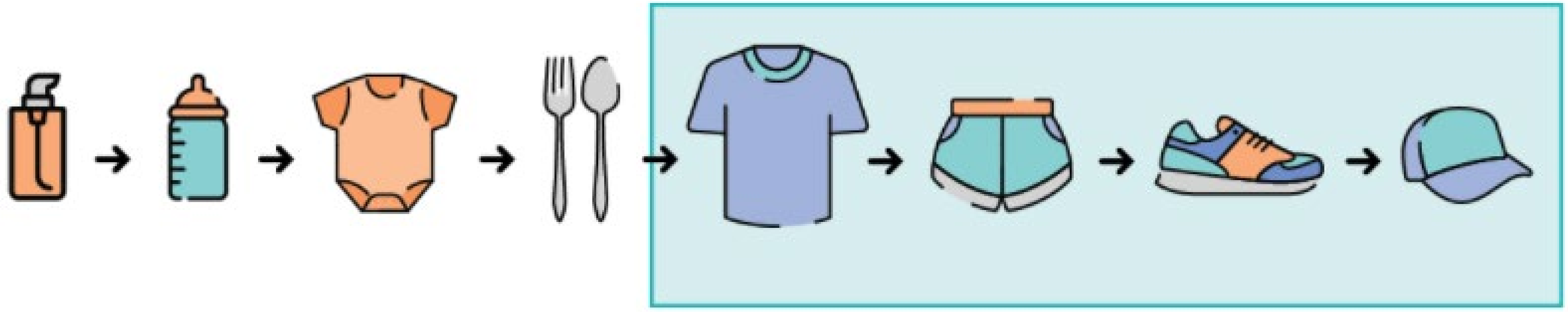
Motivations

User identification may be unknown and only the user behavior history during an ongoing session is available.

Session-based RS

Recommendation based on “*session (sequence)”*

e.g) shopping behaviors in a session

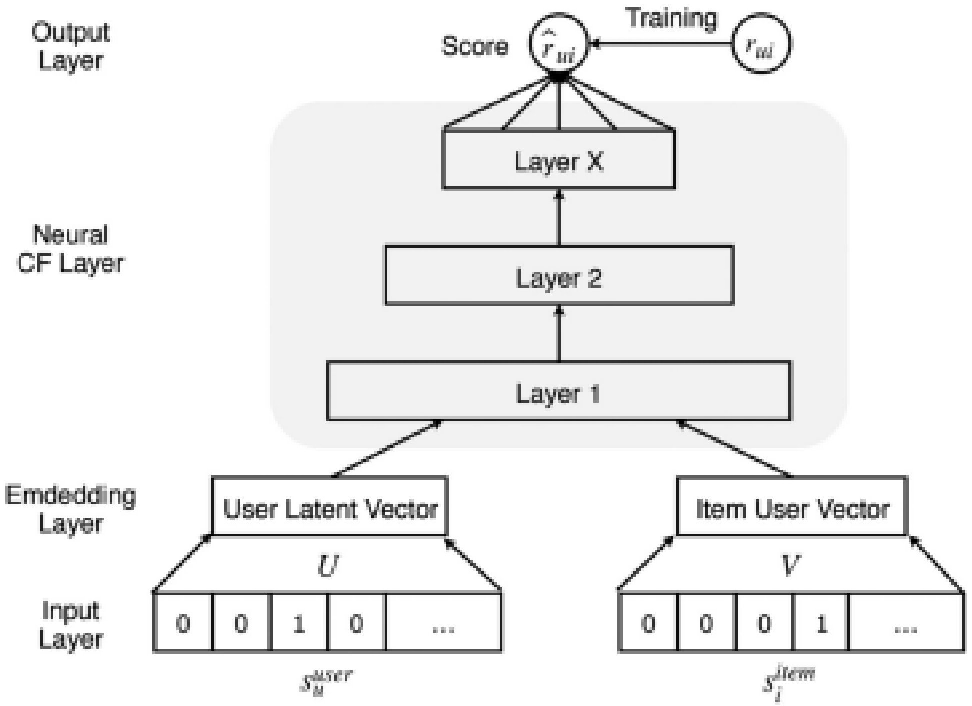


* RNN, can be a good solution.

https://blog.cloudera.com/session-based-recommender-systems/

Deep Neural Networks for Recommendation

As the influence of deep learning is getting pervasive, recently it also demonstrates effectiveness in recommender systems research.



* Neural collaborative filtering

Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." *ACM Computing Surveys (CSUR)* 52.1 (2019): 1-38.

Why Deep Neural Networks for Recommendation?

1. Nonlinear transformation

Capturing complex user/item interaction patterns

Linear model: limited modeling expressiveness

2. Representation Learning

Covering heterogeneous content information (such as text, images, audio, and even video.)

3. Sequence Modeling

CNN, RNN

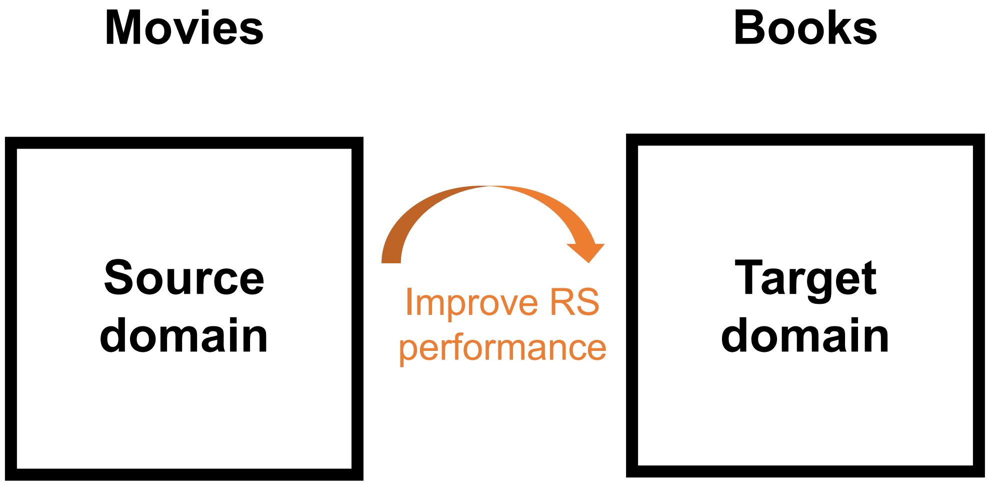
4. Flexibility

Good modulization with frameworks like TF, Keras, PyTorch, Theano, …

Data Sources/Tasks	Notes	Publications
Sequential Information	w/t User ID	[16, 29, 33, 35, 73, 91, 118, 134, 144, 161, 174, 176, 190, 195, 199, 206]
	Session based w/o User ID	[55–57, 68, 73, 100, 102, 103, 118, 143, 149, 150]
	Check-In, POI	[151, 152, 166, 186]
Text	Hash Tags	[44, 110, 119, 159, 183, 184, 194, 210]
	News	[10, 12, 113, 136, 170, 201]
	Review texts	[11, 87, 127, 147, 175, 198, 203]
	Quotes	[82, 142]
Images	Visual features	[2, 14, 25, 49, 50, 84, 99, 105, 112, 166, 173, 180, 192, 193, 198, 207]
Audio	Music	[95, 154, 168, 169]
Video	Videos	[14, 17, 27, 83]
Networks	Citation Network	[9, 38, 66]
	Social Network	[32, 116, 167]
	Cross Domain	[39, 92, 167]
Others	Cold-start	[155, 157, 171, 172]
	Multitask	[5, 73, 87, 175, 188]
	Explainability	[87, 127]

Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." *ACM Computing Surveys (CSUR)* 52.1 (2019): 1-38.

Cross-domain RS (CDRS)



Motivation

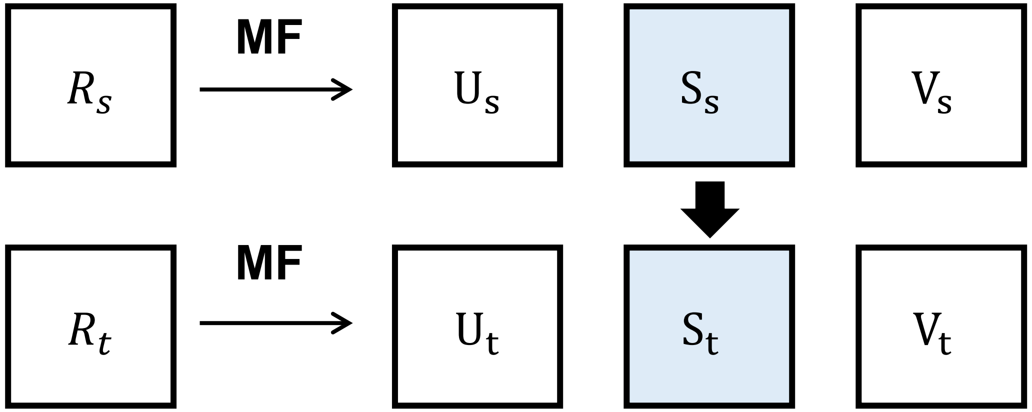
Single-domain RS suffer from sparsity and cold-start problem.

Cross-domain RS (CDRS)

- CDRS assists target domain recommendation with the knowledge learned from source domains.

- Transfer learning is most widely studied topic.

*rating matrix



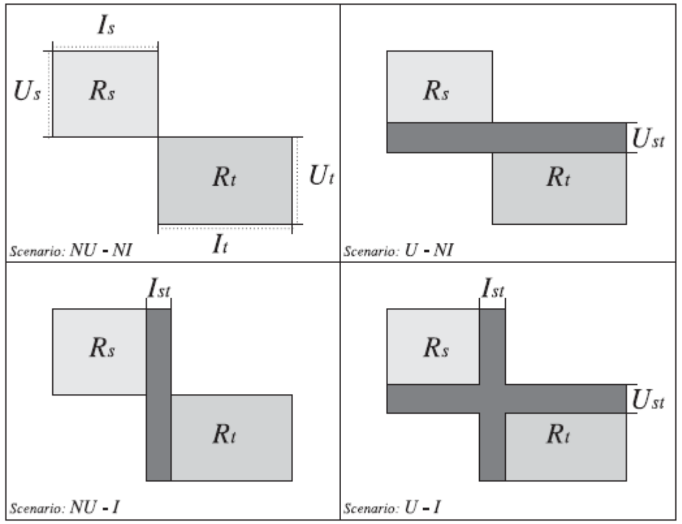
Zhang, Qian, et al. "A cross-domain recommender system with consistent information transfer." *Decision Support Systems* 104 (2017): 49-63.

Cross-domain RS (CDRS)

However, the problem definition is complicated & not clearly defined yet.

One example:
It starts from how many users & items are overlapped →

1. Not overlapped
2. Partly overlapped (items or users)
3. Fully overlapped



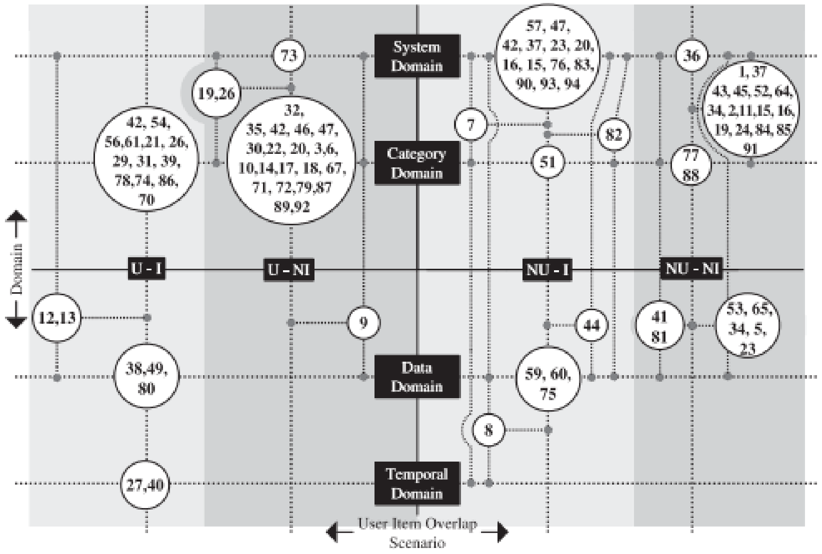


Fig. 7. Domain vs user-item overlap.

Khan, Muhammad Murad, Roliana Ibrahim, and Imran Ghani. "Cross domain recommender systems: a systematic literature review." *ACM Computing Surveys (CSUR)* 50.3 (2017): 1-34.

2nd session : SVD++, a powerful RS model

Background of SVD++

Netflix prize

NETFLIX

Oct. 2006, Netflix released
a dataset containing 100 million movie ratings and
challenged the research community to develop algorithms
that could beat the accuracy of its RS

Out of the competitive RS algorithms,
at Sep. 2009,
BellKor's Pragmatic Chaos team's
SVD++ won US$1,000,000 (the best) !

https://www.wired.com/2009/09/bellkors-pragmatic-chaos-wins-1-million-netflix-prize/

Motivation of SVD++

Two primary approaches to CF:

1) Neighborhood model

centered on computing the relationships
between items or, alternatively, between users

2) Latent factor models.

by transforming both items and users
to the same latent factor space,
thus making them directly comparable.

What **Netflix prize** teaches:

None of them is optimal on its own.

Motivation of SVD++

Two primary approaches to CF:

	1) Neighborhood model	2) Latent factor models.
pros	Effective at detecting very localized relationship	Effective at estimating overall relations
cons	Unable to capture totality	Poor at capturing strong associations among a small set of closely related items

$\underbrace{\hspace{20em}}_{\text{Combined !}}$

SVD ++

(This is the first model combining the two approaches.)

Some preliminaries

1) Feedback types

Explicit feedback:

- Explicit input by users regarding their interest in products.

e.g.) user ratings (1~5 scores), preference (thumbs-up/down button)

- Not always available

Implicit feedback

- Indirectly reflect opinion through observing user behavior

e.g.) purchase history, browsing history, search patterns, or even mouse movement.

- Relatively abundant

- * In this paper, It indicates just whether he/she saw the movie or not

Some preliminaries

2) Major notations

- u, v : users // i, j : items
- r_{ui} : known ratings (1~5)
- $\hat{r}_{ui}$: predicted ratings (1~5)

  * usually, vast majority of ratings are unknown
  * for Netflix, 99% ratings are missing (very sparse)

- $R(u)$: set of items that rated by user u
- $N(u)$: set of items that implicit preference is given by user u

SVD++

Combination of the three major parts !

1) Baseline estimation

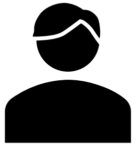
2) Neighborhood model

3) Latent factor model

1) Baseline estimates

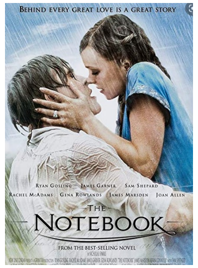
There is a *rating tendency* in both user and item.

e.g.) Two people watched a same movie and felt same impression,
but give different ratings.



Joe

Watched →

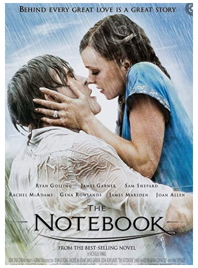


→ 4.9

Joe tends to give high ratings

Clair

Watched →



→ 4.0

Clair tends to give low ratings

1) Baseline estimates

- For reflecting **systematic tendencies** for some users to give higher ratings than others, and for some items to receive higher ratings than others

$$b_{ui} = \mu + b_u + b_i$$

overall
average rating
(known)

observed
deviations of
user
(to be found)

observed
deviations of
item
(to be found)

1) Baseline estimates

- To find b_u and b_i, solve this least square problem with given r_{ui}

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2\right)$$

The **regularizing term** to avoids overfitting by penalizing the magnitudes of the parameters.

2) Neighborhood model

1. Item similarity calculation

There are other suggestions for item similarity measure,
but this is one of the typical.

$$s_{ij} \stackrel{\text{def}}{=} \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij}$$

- n_{ij}: number of users that rated both items i,j
- ρ_{ij}: Pearson correlation coefficient (measuring the tendency of users to rate items similarly)
- λ: hyperparameter (usually, set near 100)

We can extract $S^k(i)$, set of top-k similar items based on this.

2) Neighborhood model

2. History of SVD++ neighborhood model

1. Personalized weight -> global weight w_{ij}

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) \boxed{w_{ij}}$$

2. Emphasizing implicit feedback

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + \boxed{\sum_{j \in \mathrm{N}(u)} c_{ij}}$$

2. Increase the influence of top-k similar items

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{\boxed{j \in \mathrm{R}^k(i;u)}} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{\boxed{j \in \mathrm{N}^k(i;u)}} c_{ij}$$

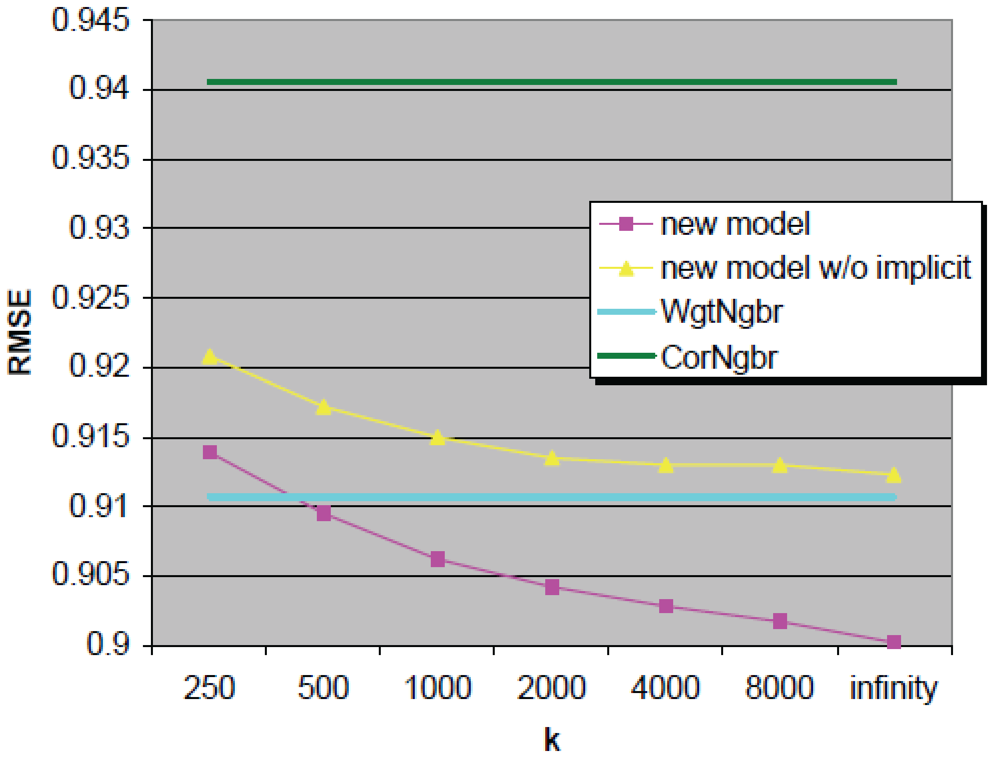
$$\mathrm{R}^k(i;u) \stackrel{\text{def}}{=} \mathrm{R}(u) \cap \mathrm{S}^k(i)$$
$$\mathrm{N}^k(i;u) \stackrel{\text{def}}{=} \mathrm{N}(u) \cap \mathrm{S}^k(i).$$

(* Please remember this formulation, even vaguely.)

2) Neighborhood model

Interim results with only neighborhood model

(* RMSE: Root mean square error)



3) Latent factor model

*** SVD models have gained popularity, thanks to its accuracy and scalability.**

$$R = M\Sigma U^T$$

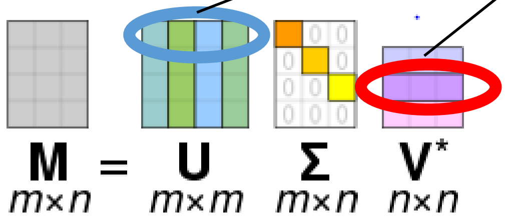
Here, decompose rating matrix R, to predict rating values for missing components.

3) Latent factor model

*** SVD-based RS models**

The prediction is done by taking an inner product of item-factor vectors p_i and user-factor vectors p_u.

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i$$



Here, decompose rating matrix R, to get latent vectors p_i and p_u.

3) Latent factor model

But wait !

SVD can be conducted on complete matrix ..

- Initial approaches: imputation based.
(e.g. replacing missing values with mean rating)
-> poor performance.

* Most of the rating matrix data has high portion of missing values.
Ex) in the Netflix data **99%** of the possible ratings are missing.

3) Latent factor model

SVD can be conducted on complete matrix.

Thus, **SVD++ is not a SVD**, precisely speaking.

Instead, is converted to a **minimization problem on the *known* *ratings*.**

$$r_{ui} = b_{ui} + p_u^T q_i$$

Known ratings

Representations for users and items

3) Latent factor (LF) model

SVD++'s LF: "Asymmetric-SVD"

Replace p_u
to representation with the items they prefer

$$\hat{r}_{ui} = b_{ui} + \boxed{p_u^T} q_i$$

$$\hat{r}_{ui} = b_{ui} + q_i^T \boxed{\left(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) x_j + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)}$$

What are the benefits of this?

1. Fewer parameters
Since # items << # users, usually.

2. New users
Practically, systems need to provide immediate recommendations to new users who expect quality service

3. Explainability

4. Integration of implicit feedback
$\mathrm{N(u)}$ term

3) Latent factor model

Optimization for the model

$$\boxed{\min_{q_*,x_*,y_*,b_*} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - q_i^T \left(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j\in\mathrm{R}(u)} (r_{uj} - b_{uj})x_j + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j\in\mathrm{N}(u)} y_j \right) \right)^2}$$

$$\boxed{+ \lambda_5 \left(b_u^2 + b_i^2 + \|q_i\|^2 + \sum_{j\in\mathrm{R}(u)} \|x_j\|^2 + \sum_{j\in\mathrm{N}(u)} \|y_j\|^2 \right)}$$

Again, just a combination of (Least square problem + regularizer)

SVD++, an integrated model

Rating prediction of SVD++

2) Asymmetric SVD

1) Baseline estimation

$$\hat{r}_{ui} = \boxed{\mu + b_u + b_i} + \boxed{q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)}$$
$$+ \boxed{|\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}}$$

3) Neighborhood based

And its parameters are updated, accordingly by gradient descent

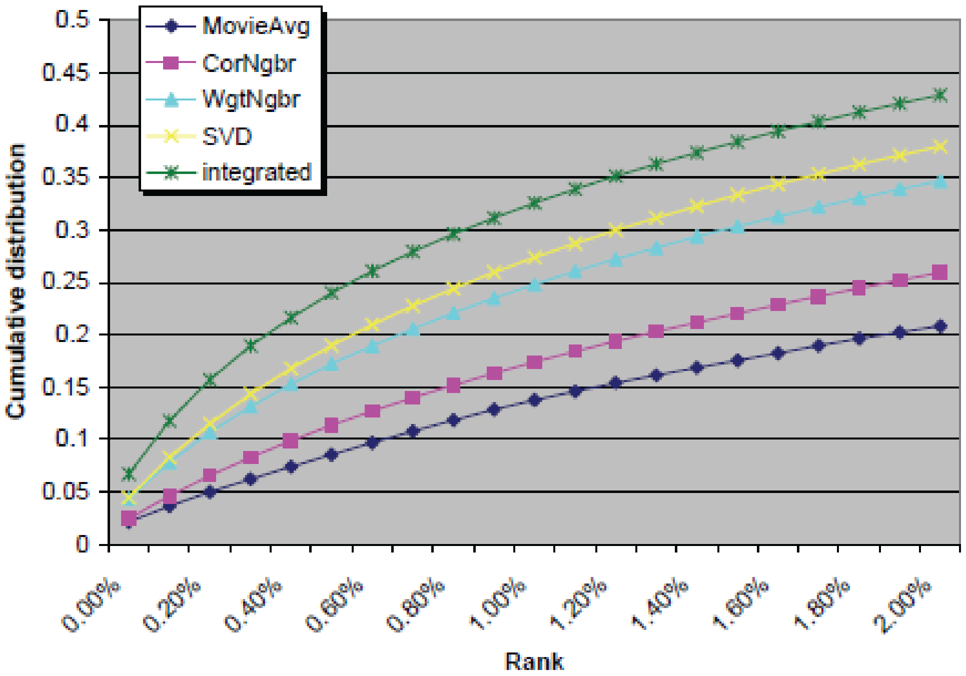
- $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$
- $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j) - \lambda_7 \cdot q_i)$
- $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$
- $\forall j \in \mathrm{N}(u)$:
  $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |\mathrm{N}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_7 \cdot y_j)$
- $\forall j \in \mathrm{R}^k(i;u)$:
  $w_{ij} \leftarrow w_{ij} + \gamma_3 \cdot \left(|\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_8 \cdot w_{ij} \right)$
- $\forall j \in \mathrm{N}^k(i;u)$:
  $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot \left(|\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_8 \cdot c_{ij} \right)$

SVD++, an integrated model

Conventional →
Only latent factor model →
Integrated →

Model	50 factors	100 factors	200 factors
SVD	0.9046	0.9025	0.9009
Asymmetric-SVD	0.9037	0.9013	0.9000
SVD++	0.8952	0.8924	0.8911

Cumulative distribution of the correct case (inferring 5 star ratings)



Either Neighborhood model and latent factor model cannot win the integrated model

References

- Bobadilla, Jesús, et al. "Recommender systems survey." *Knowledge-based systems* 46 (2013): 109-132.

- Sridevi, M., R. Rajeshwara Rao, and M. Varaprasad Rao. "A survey on recommender system." *International Journal of Computer Science and Information Security* 14.5 (2016): 265.

- Khan, Muhammad Murad, Roliana Ibrahim, and Imran Ghani. "Cross domain recommender systems: a systematic literature review." *ACM Computing Surveys (CSUR)* 50.3 (2017): 1-34.

- Tang, Jie, et al. "Cross-domain collaboration recommendation." *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012.

- Lu, Jie, et al. "Recommender system application developments: a survey." *Decision Support Systems* 74 (2015): 12-32.

- Si, Luo, and Rong Jin. "Flexible mixture model for collaborative filtering." *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003.

- Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*.
- 2008.Ding, Chris, et al. "Orthogonal nonnegative matrix t-factorizations for clustering." *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006.

- Youtube, 허민석, 추천 시스템 기본 - (콜라보레이티브 필터링, 컨텐트 베이스 필터링) https://www.youtube.com/watch?v=_YndKkun2Sw

- Youtube . Computing for all, 2.4 Data Science: Jaccard Coefficient or Index or Similarity. https://www.youtube.com/watch?v=HMx-lGSkcwM

- Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." *ACM Computing Surveys (CSUR)* 52.1 (2019): 1-38.

Thanks for your listening.