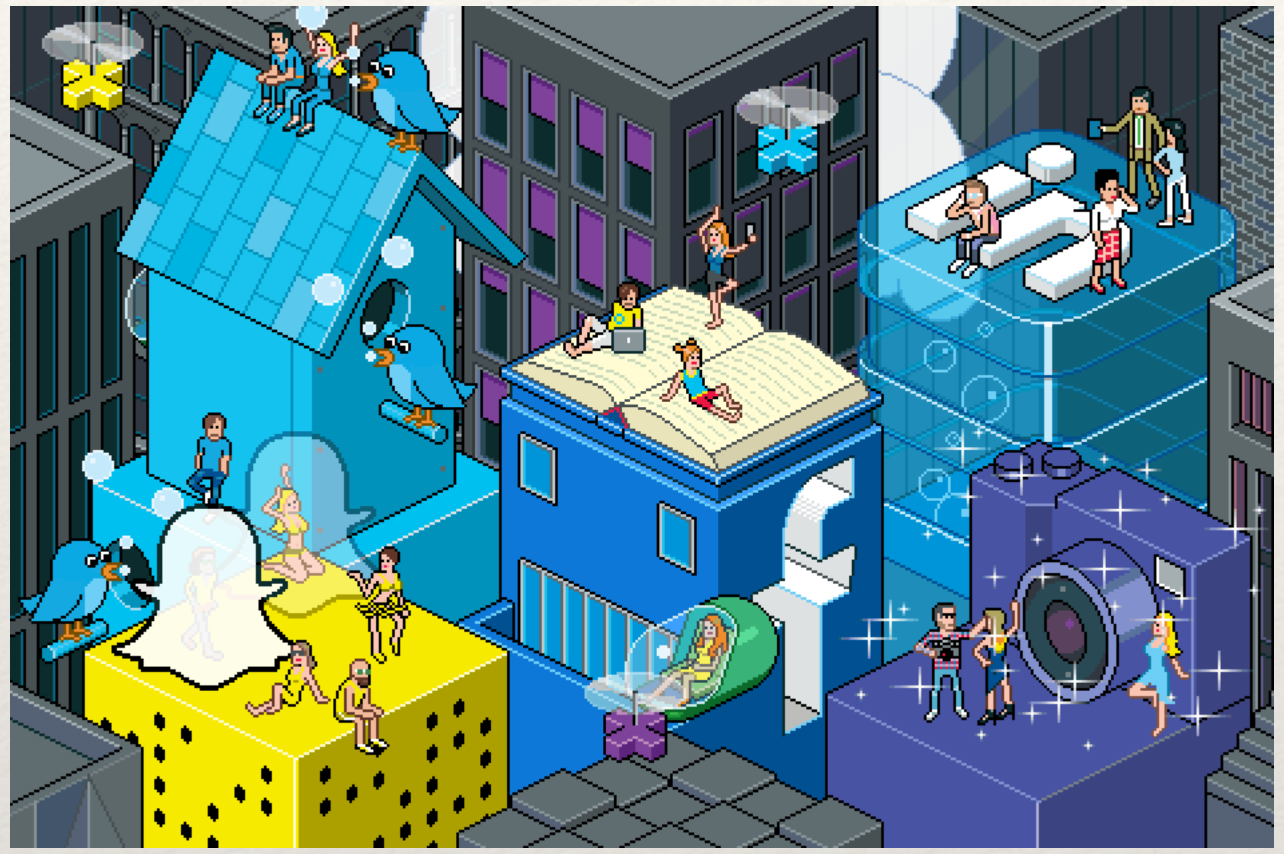
Intro to Recommendation Systems

Segev Arbiv

segevarbiv@gmail.com

Basic idea - motivation



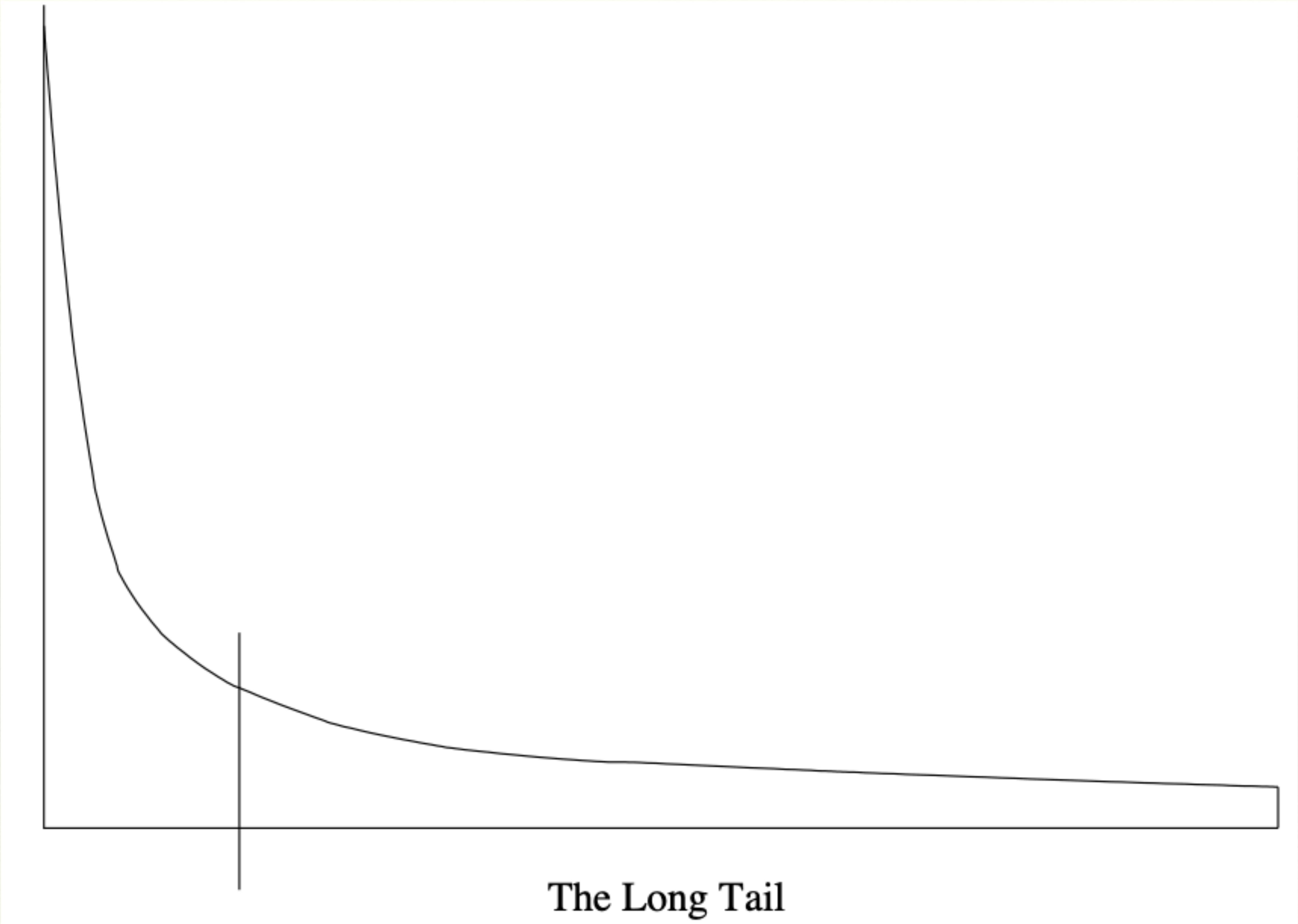
https://tenor.com/view/social-media-community-snapchat-picture-twitter-gif-13718043

https://www.slideshare.net/mdeiters/you-might-also-like-implementing-user-recommendations-in-

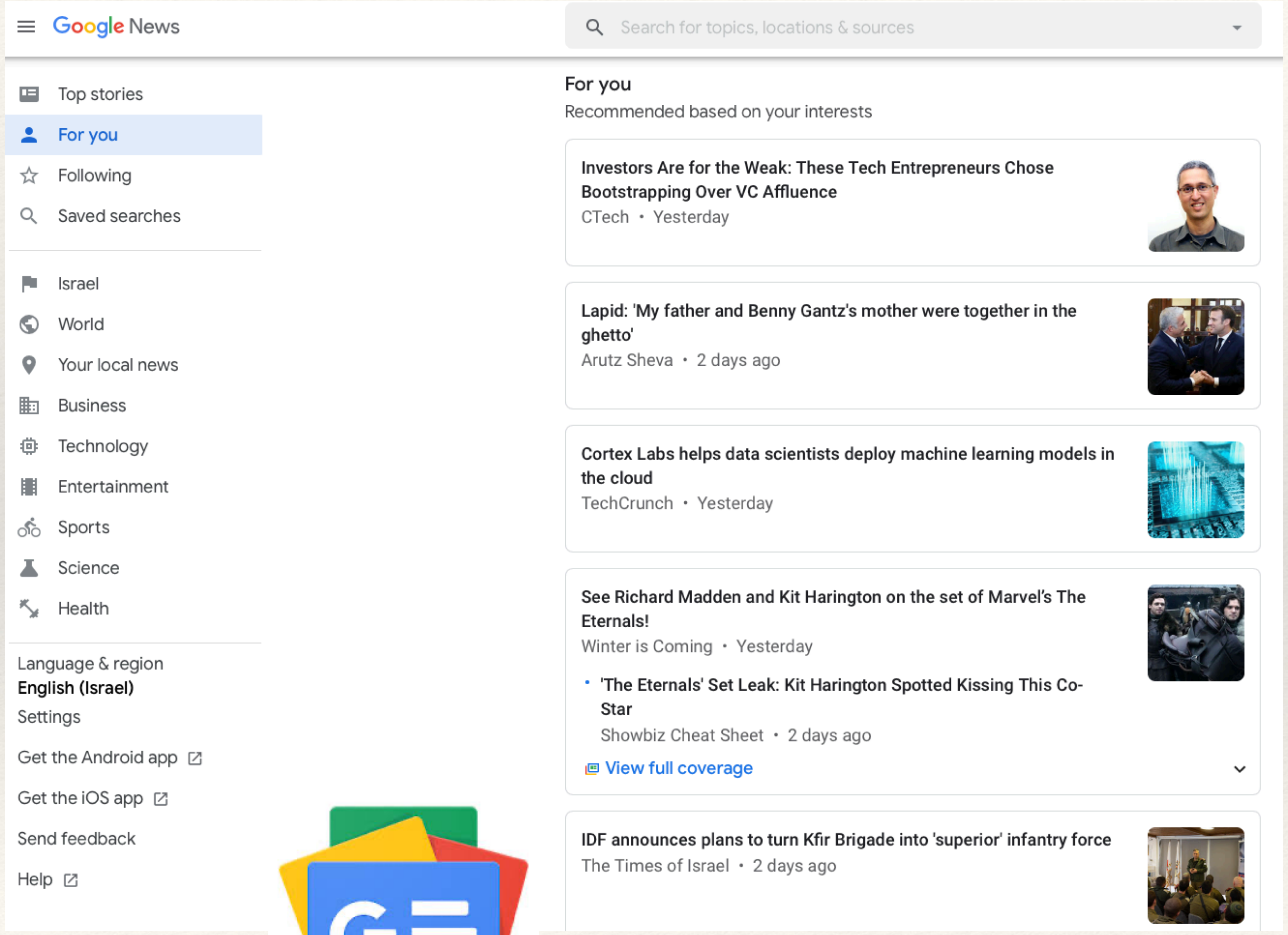
By Moshanin - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24097346

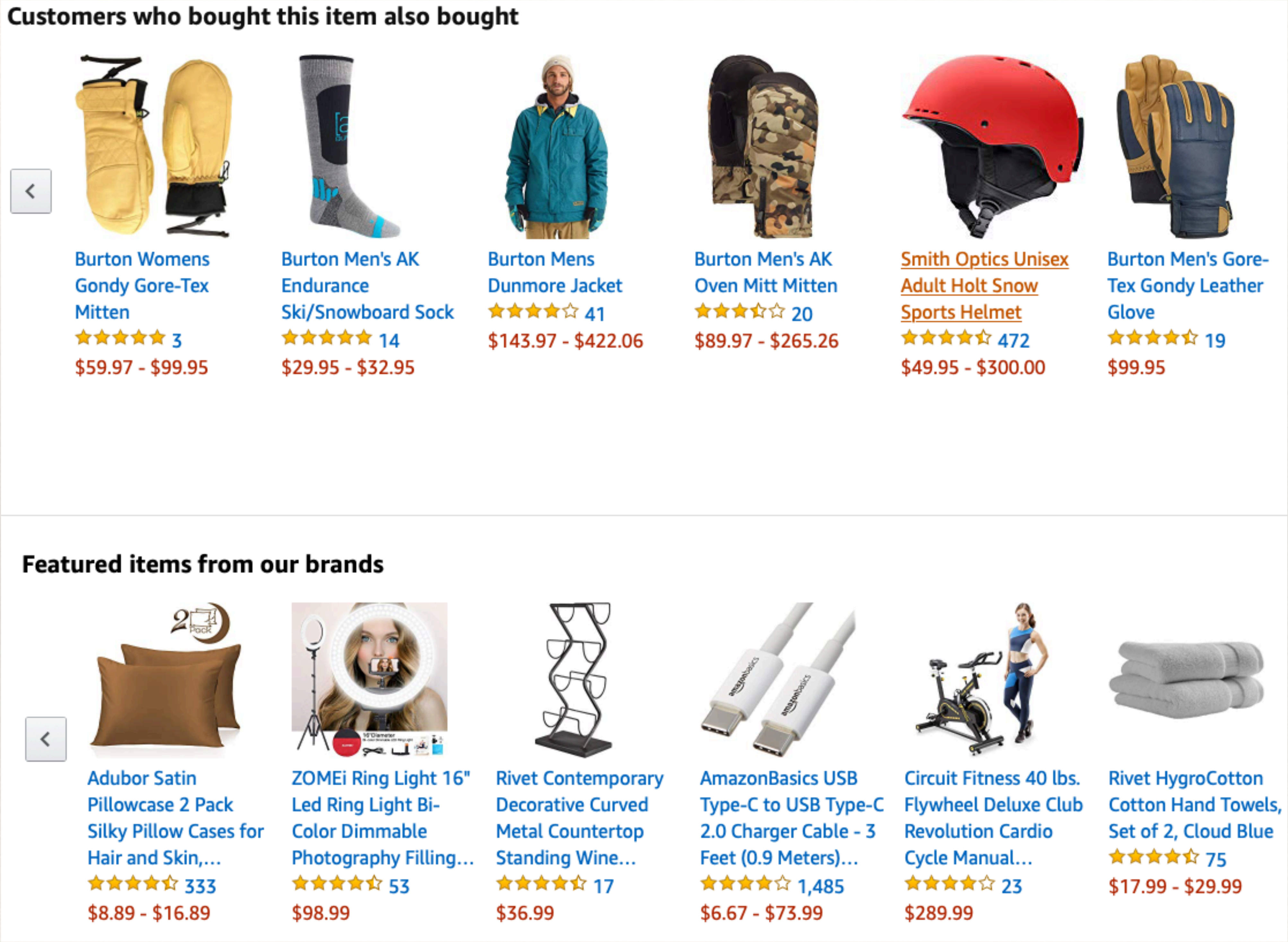
Long tail - motivation

- Shelf space of retail vs commerce
  - Retailers can only present the most popular items due to lack of space (left of vertical line)
  - e-Commerce allow us to have tons of info about ~infinite products.
  - more choices calls for better recommendation engines…

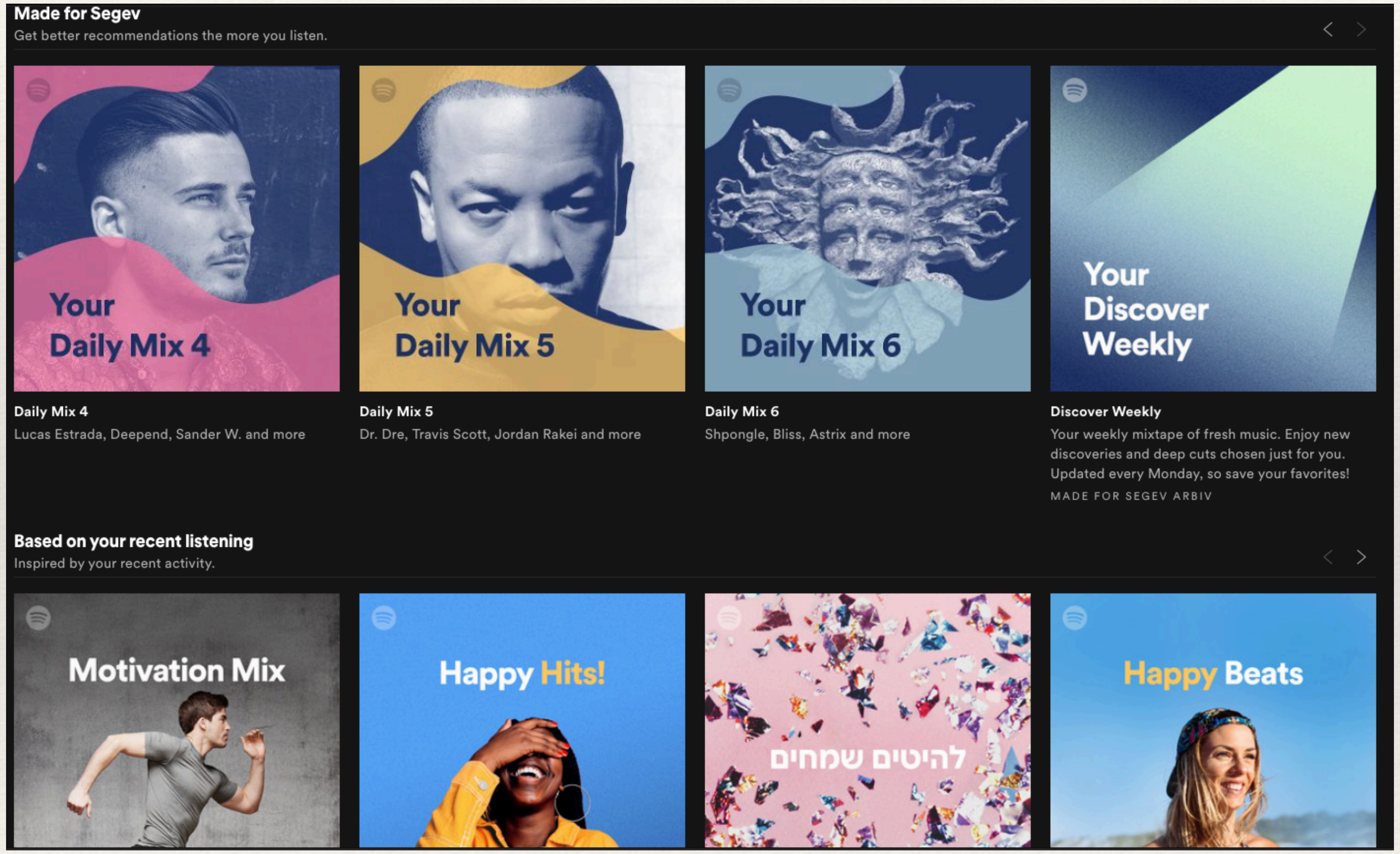


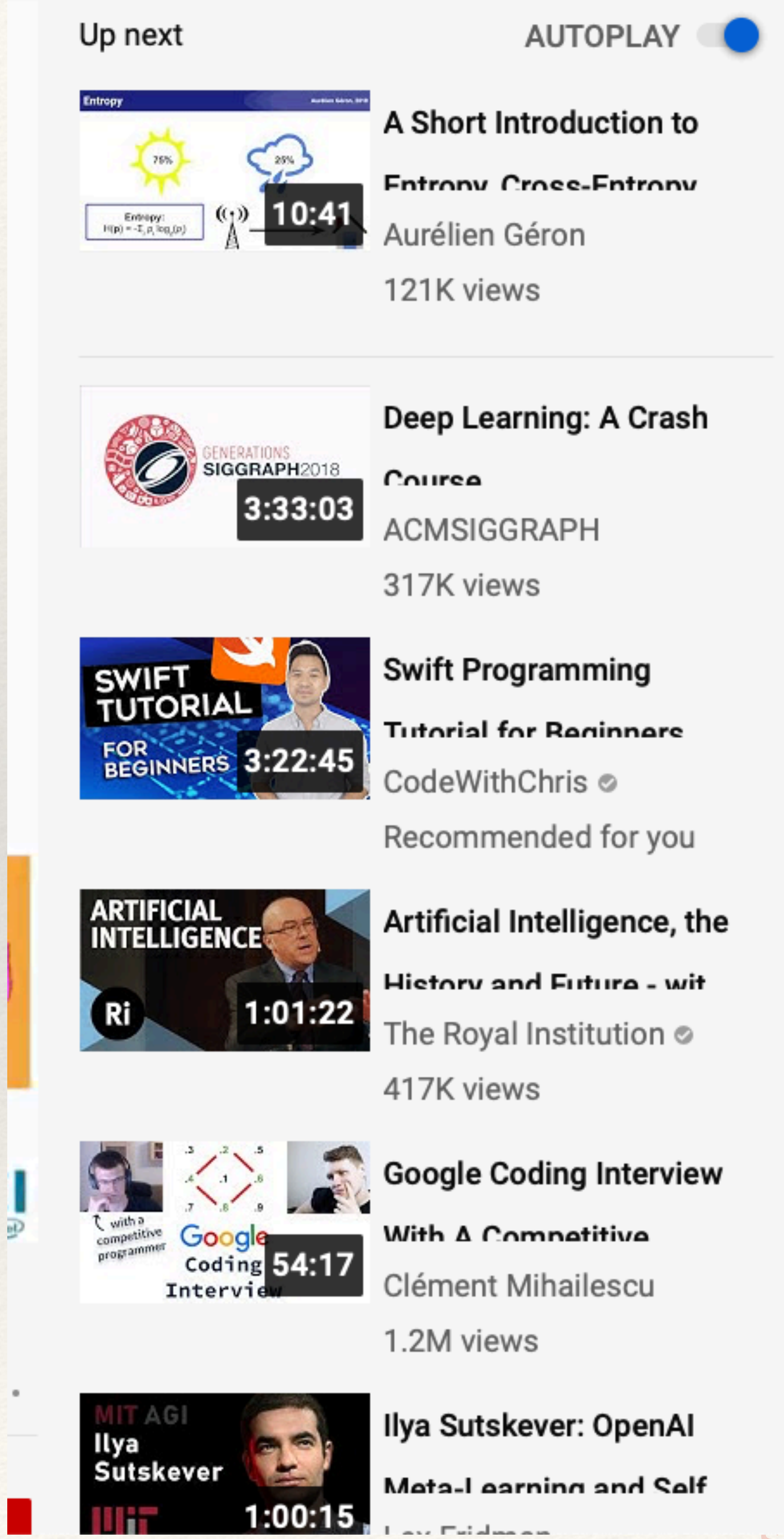
Common use cases





Common use cases





Common use cases

- Netflix Prize
  - 100M ratings
  - 480K users
  - ~18K movies
    - 1.1% observed data…
  - Test set of 2.8M ratings
  - RMSE

Netflix Prize

Problem definition

Problem definition

- ***U*** - a set of users
- ***I*** - a set of items
- ***R*** - utility\ranking matrix
    - r_{ui} - the rating user u gave item i
        - #⭐, 👍\👎, real numbers in [0,1] etc.

Di Noia, Tommaso & Ostuni, Vito. (2015). Recommender Systems and Linked Open Data

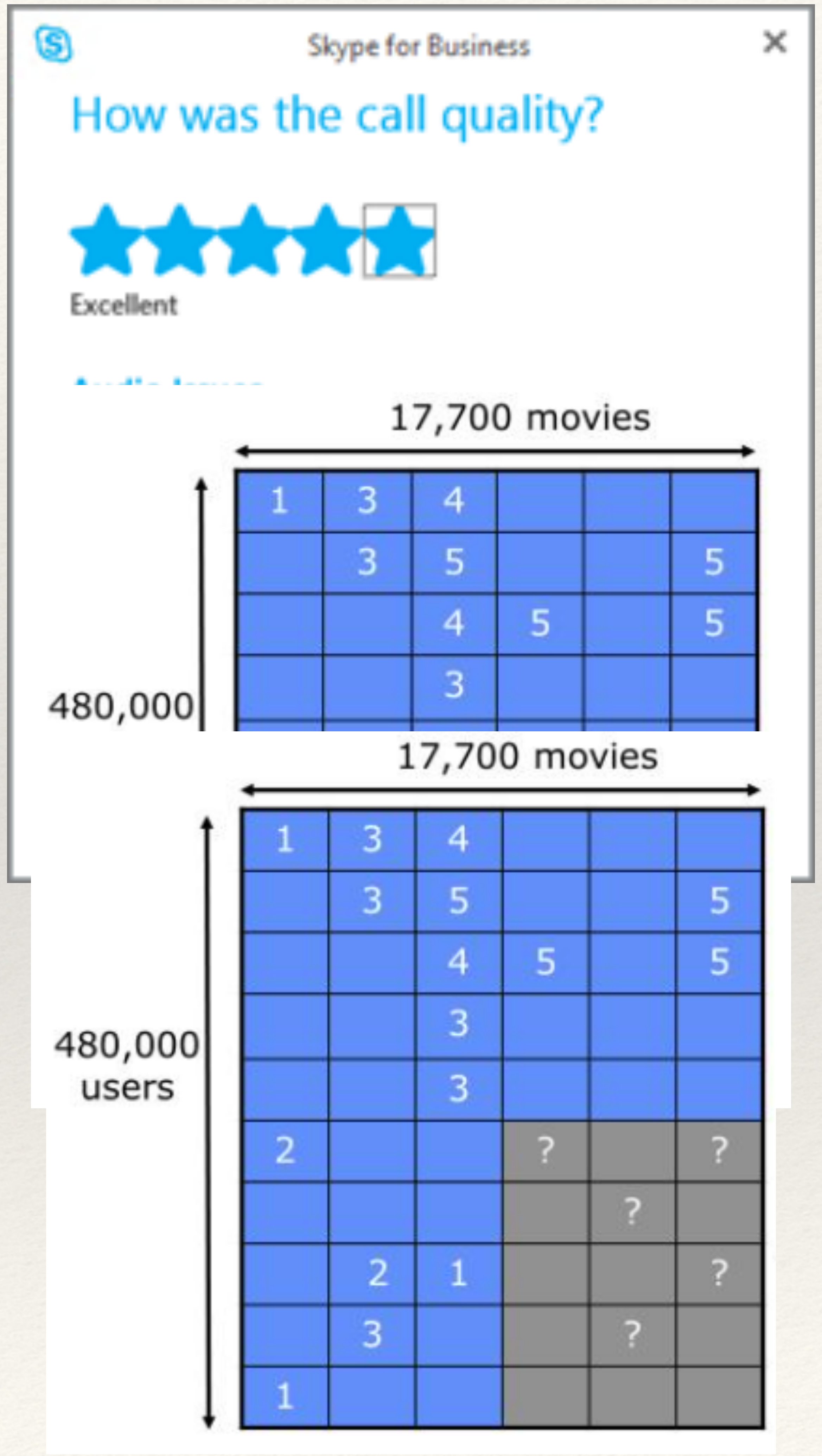
Our goal - fill in the missing entries.

Problem definition
Key challenges

- How to get the “known” ratings?
  - Users are often unwilling to rate a product.

- Rating matrix is very spares
  - Long tail of items have not been rated
  - Suggest topN instead of rate all items

- Evaluation
  - How to evaluate a model on unseen items?

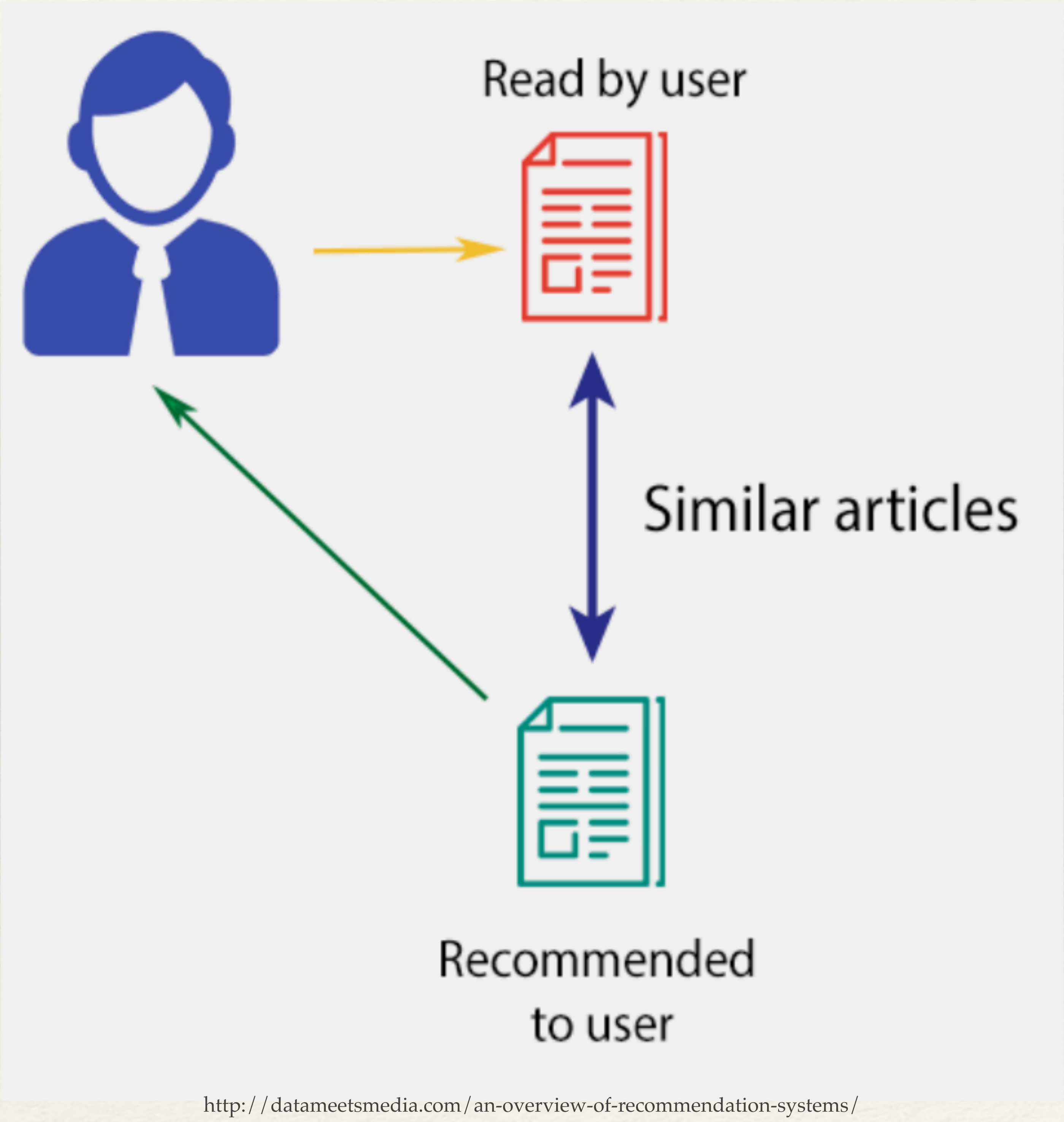


CS 277: THE NETFLIX PRIZE PROFESSOR PADHRAIC SMYTH

Content Based Recommendation

Content Based Recommendation

- **Main idea**:
  - Use item features to recommend similar items to what the users likes, based on their previous actions or explicit feedback.



http://datameetsmedia.com/an-overview-of-recommendation-systems/

- For example:
  - Recommend movies from same director
  - Recommend blogs with "similar" content
  - Recommend music from the same genre.

Content Based Recommendation
Profiling an item

- A feature vector per item should be constructed. Features can be both numerical and categorical:
  - Movies: Actors, title, director, tags…
  - Music: length, genre, artist…
  - Documents\books: TF-IDF, NN based features (doc2vec, transformers)…

Content Based Recommendation
Profiling an user

- We should represent user in the same feature space as the items.
    - (weighted) Average of rated item features.
    - **Explicit** and **implicit**
        - Selected *Comedy* as a favourite genre
        - Bought items from the same company

	Comedy	Romance	…	Action
MARVEL STUDIOS AVENGERS ENDGAME	0.5	0.2	…	0.9
DUMB AND DUMBER TO	1	0.2	…	0.2
TITANIC	0	0.8	…	0.3
	0.5	0.4	…	0.46

Content Based Recommendation
Prediction

- Given user profile and item profile, rating can be computed:
    - dot product: $\langle u, i \rangle = \sum_{m=1}^{d} u_m i_m$
    - cosine similarity: $\cos(\theta_{ui}) = \frac{\langle u, i \rangle}{||u|| \cdot ||i||}$
    - etc.
- Thats it. Simple as that.
    - Often lead to poor results…

Content Based Recommendation

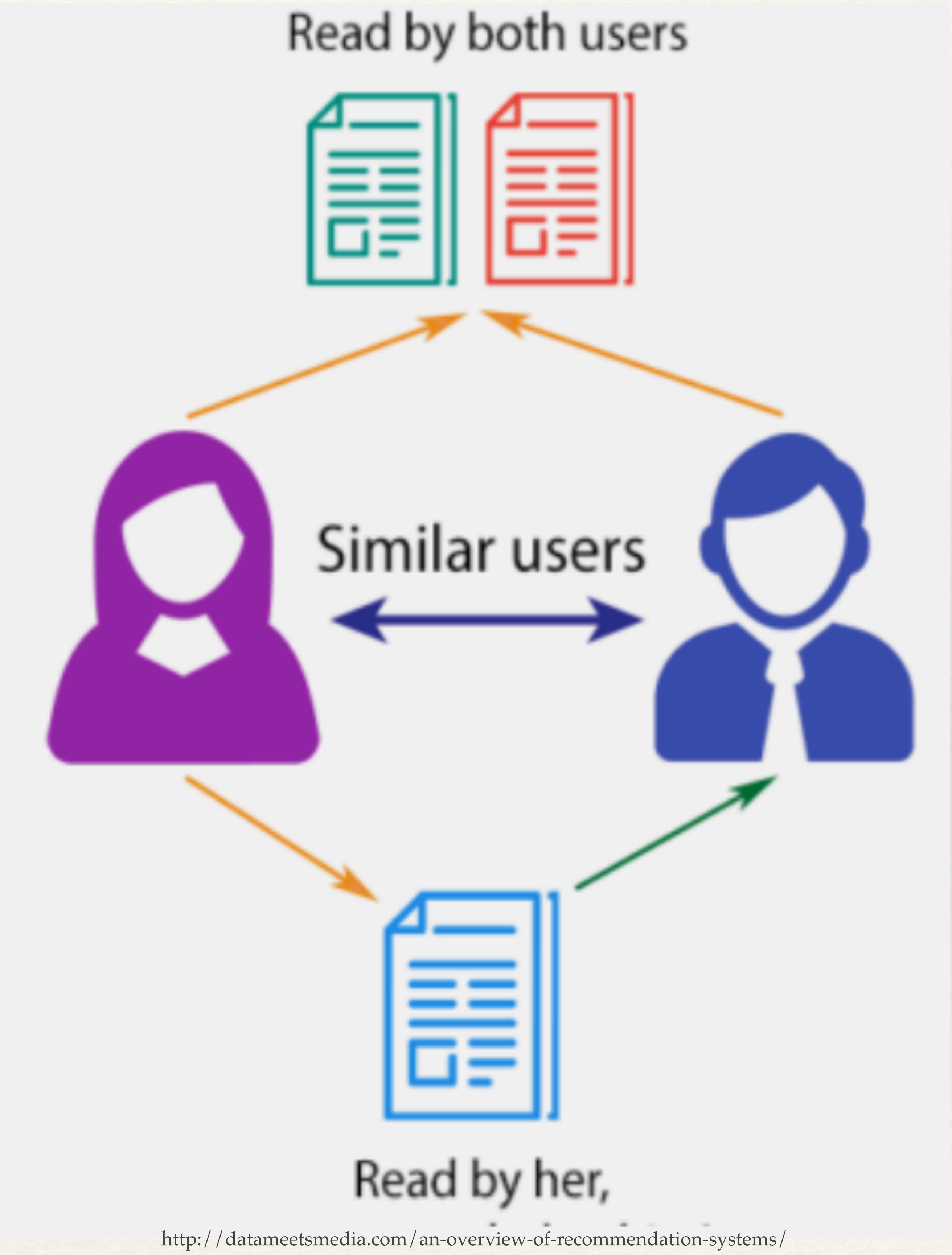
- Pros
  - No need for data on other users - easier to scale.
  - capture the specific interests of a user —> can recommend niche items.
  - Easily explainable - list of content-features that caused recommendation.
- Cons
  - Complicated feature engineering - requires domain expertise.
  - Model has limited ability to expand on the users' existing interests.
  - Cold start for new users.

Collaborative Filtering

Collaborative Filtering

- **Main idea:**
  - Given user u
  - Find similar users <—> "similar" rating of items.
  - Estimate u's rating based on the similar users.

- Doesn't rely on user\item attributes, which are usually missing or hard to compute.



http://datameetsmedia.com/an-overview-of-recommendation-systems/

Collaborative Filtering
Similarity

$r_u = [\,*\,,\ \_,\ \_,\ \ *\,,\ ***\,]$

$r_v = [\,*\,,\ \_,\ \ **,\ \ **\,,\ \_]$

Let r_u be the rating vector of user u

- **Jaccard index (IoU)**

$$J(r_u, r_v) = \frac{|r_u \cap r_v|}{|r_u \cup r_v|}$$

  - problem - Ignores the value of the rating

$$\boxed{\begin{array}{l} r_u = \{1,4,5\} \\ r_v = \{1,3,4\} \end{array}}$$

- **Cosine similarity**

$$sim(u, v) = \cos(\theta_{r_u, r_v}) = \frac{\langle r_u, r_v \rangle}{\|r_u\| \cdot \|r_v\|}$$

  - problem - Treat missing values as 0's (thus, "negative")

$$\boxed{\begin{array}{l} r_u = [1,0,0,1,3] \\ r_v = [1,0,2,2,0] \end{array}}$$

- **Pearson correlation**

$$sim(u, v) = \frac{\sum_{s \in S_{uv}} (r_{us} - \bar{r}_u)(r_{vs} - \bar{r}_v)}{\sqrt{\sum_{s \in S_{uv}} (r_{us} - \bar{r}_u)^2} \sqrt{\sum_{s \in S_{uv}} (r_{vs} - \bar{r}_v)^2}}$$

  - S_{uv} - items that both users rated

Collaborative Filtering
Similarity - example

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Intuitively, $sim(A, B) > sim(A, C)$
  - Jaccard:
    $$sim(A, B) = \frac{1}{5} < \frac{2}{4} = sim(A, C)$$
  - Cosine:
    $$sim(A, B) = \frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{5^2 + 5^2 + 4^2}} = 0.38$$
    $$sim(A, C) = \ldots\ldots\ldots\ldots\ldots = 0.322$$
- Cosine indeed gives better similarity in this case, but not by a knockout…

Collaborative Filtering
Similarity - example

- By removing the mean rating of each user - we treat low rating as negative:
  - Cosine:
    $sim(A, B) = 0.092 > -0.559 = sim(A, C)$
- Notice, in this case cosine is equivalent to Pearson correlation.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Collaborative Filtering
User based

- From similarity to rating estimation:

  $\hat{r}_{ui}$ - estimation for how user u will rate item i

  - $\hat{r}_{ui} = \frac{1}{N} \sum_{v} r_{vi}$
  - $\hat{r}_{ui} = \frac{\sum_{v} sim(u, v) \cdot r_{vi}}{\sum_{v} sim(u, v)}$

?

Collaborative Filtering
User based with neighbourhood

- **Problem**:
  - massive tail of non-similar users affect our estimation for rating item *i*.
- **Solution**:
  - Take only top *k* similar users into consideration (who also rated item *i*).
- further reading: User optimized weights
  - Improved Neighborhood-based Collaborative Filtering, Bell and Koren, 2007

$$\hat{r}_{ui} = \frac{1}{N} \sum_{v} r_{vi}$$

$$\downarrow$$

$$\hat{r}_{ui} = \frac{1}{k} \sum_{v \in N(u)} r_{vi}$$

Collaborative Filtering
Baseline predictor

- **Problem**: Some users are more "generous" than others, and some items systematically receive higher ratings
- **Solution**: centre the data and add a bias term

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

  - μ - average rating of the overall items
  - b_u - rating deviation of user u from μ
  - b_i - rating deviation of item i from μ

- Mean movie rating: **3.7 stars**
- *The Sixth Sense* is **0.5** stars above avg.
- Joe rates **0.2** stars below avg.

  ⇒ **Baseline estimation:**

  ***Joe* will rate *The Sixth Sense* 4 stars**

Jure Leskovec, Stanford C246: Mining Massive Datasets

Collaborative Filtering
User based

- Estimation process for user u and item i:
  - Define a similarity measurement: $sim(u, v)$
  - Select k nearest neighbours of u: $N_k(u)$
  - Estimate rating r_{ui} as follows:

$$r_{ui} = b_{ui} + \frac{\sum_{v \in N_k(u)} sim(u, v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_k(u)} sim(u, v)}$$

Collaborative Filtering
Item based

- We can also look at the problem from an *item based* perspective.
- The approach is the same, and tends to **work better** due to users complex and versatile taste.

- Estimation flow for user u and item i:
  - Define a similarity measurement: $sim(i, j)$
  - Select k nearest neighbours of i: $N_k(i; u)$
    - items most similar to i, that were rated by u
  - Estimate rating r_{ui} as follows:

$$r_{ui} = b_{ui} + \frac{\sum_{j \in N_k(i;u)} sim(i, j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_k(i;u)} sim(i, j)}$$

Collaborative Filtering
Item based (w/o baseline example)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Jure Leskovec, Stanford C246: Mining Massive Datasets

Collaborative Filtering
Item based (w/o baseline example)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 3.6$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute cosine similarities between rows

Jure Leskovec, Stanford C246: Mining Massive Datasets

Collaborative Filtering
Item based (w/o baseline example)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

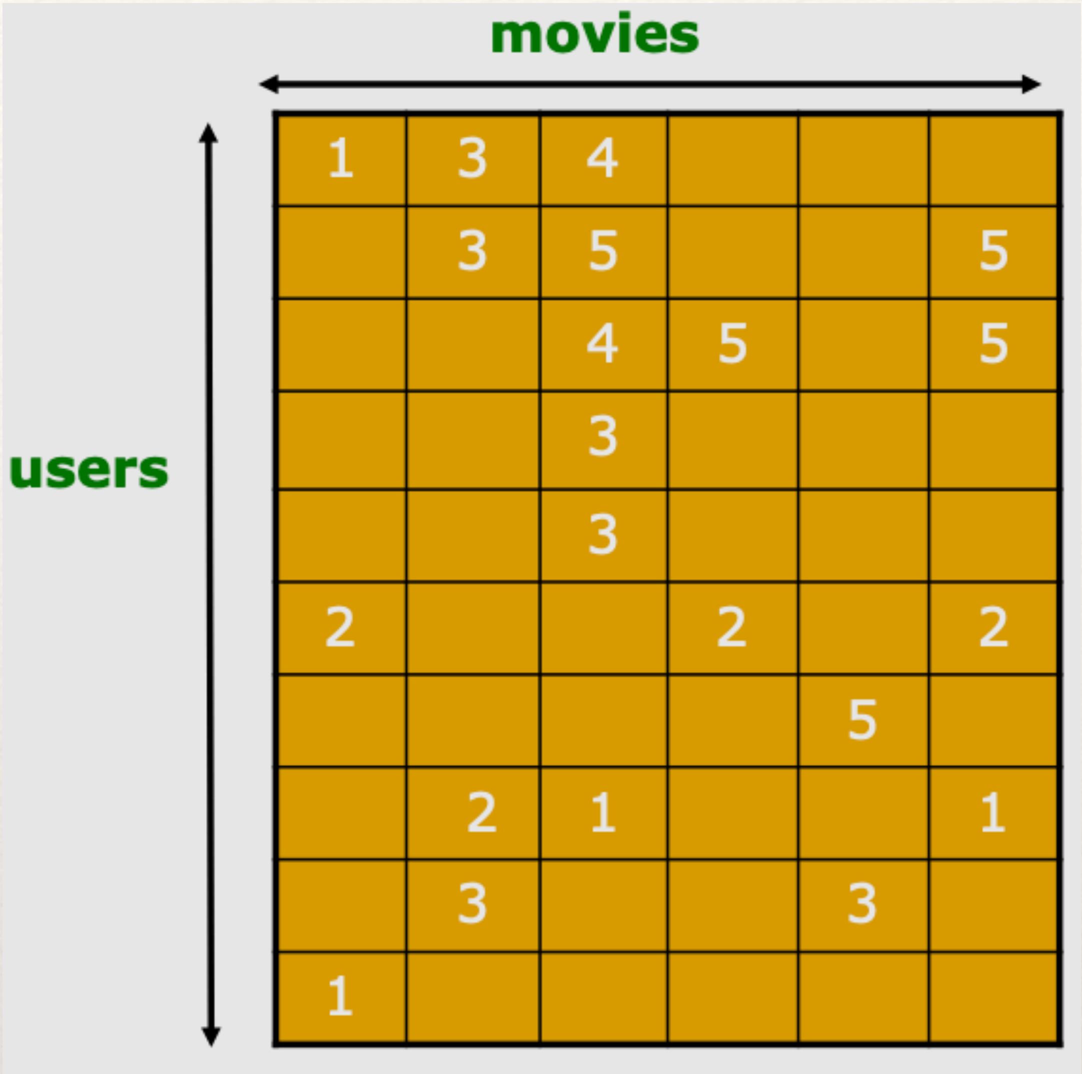
$r_{1.5} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

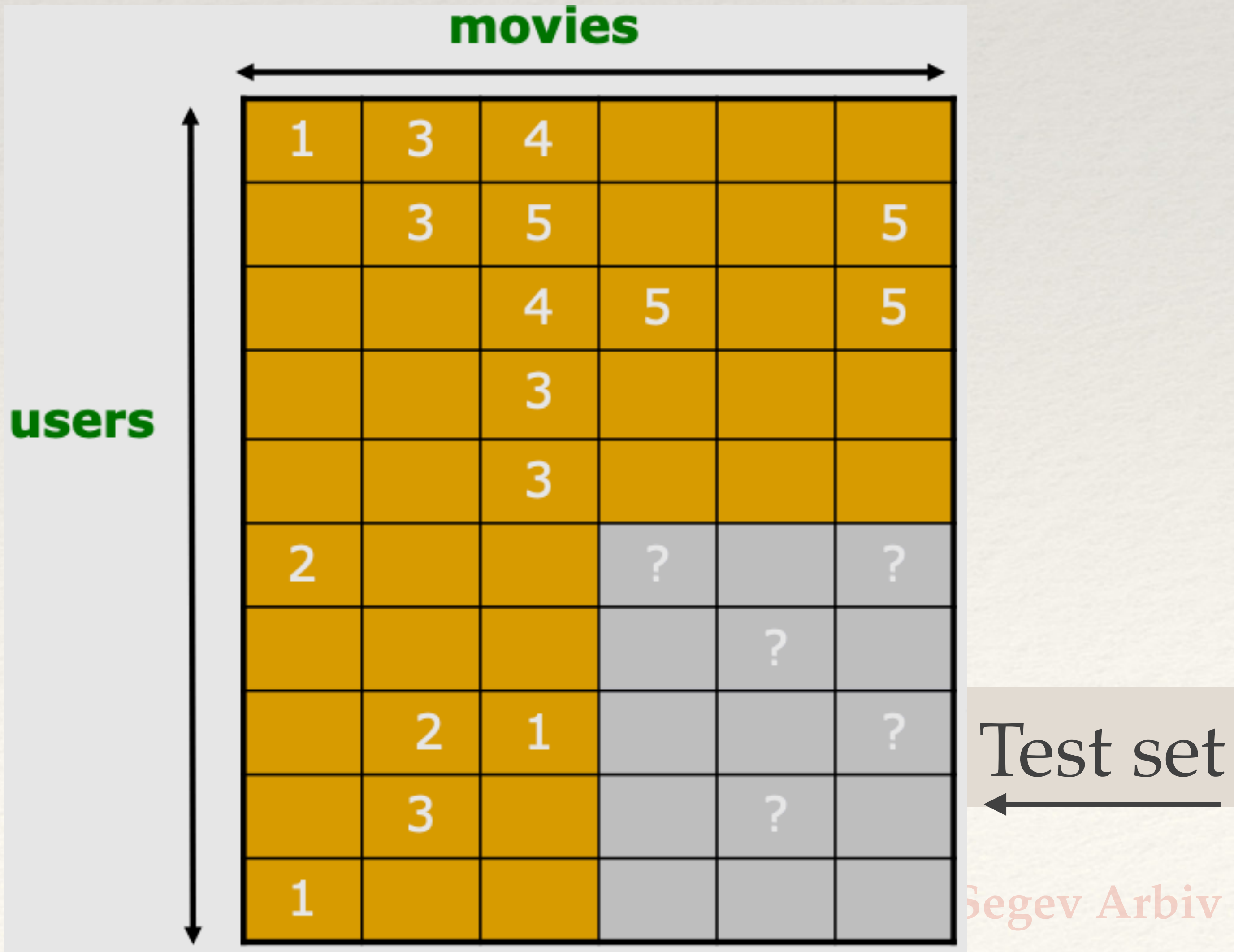
$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

Jure Leskovec, Stanford C246: Mining Massive Datasets

Collaborative Filtering
Evaluation

- On a test set Q, perform on known ratings:
  - $$RMSE = \sqrt{\frac{\sum_{(u,i)\in Q}(r_{ui} - \hat{r}_{ui})^2}{|Q|}}$$
  - P@k - What fraction of true top-k preferences are in predicted top k?
  - Ranking: spearman correlation between system and user rankings.





Collaborative Filtering

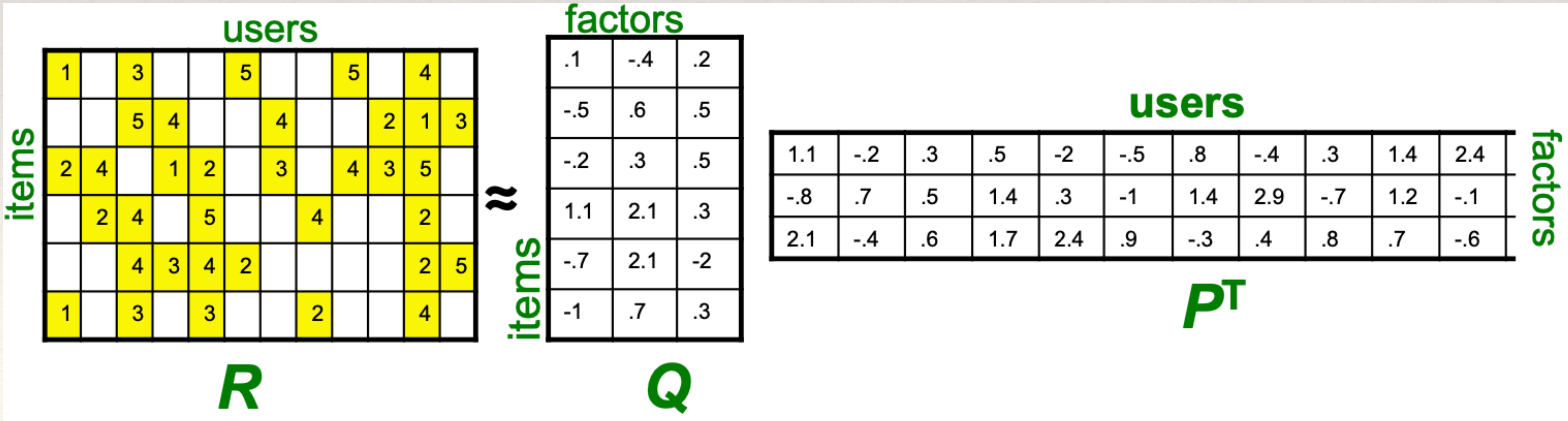
- Pros
  - No need for hand crafter feature engineering
  - More explorative that content-based
- Cons
  - Cold start - need enough users to rate an item
  - Can't recommend an item that hasn't been rated yet
  - Sparsity

- **Hybrid models** combine content-based features in the above method.

Model based - matrix factorization
(Latent factor model)

Model based - matrix factorization

- **Main idea:**
  - Overcoming the sparsity and scalability by moving to *low-dim* matrices, that also exploit *user-item* interactions.
  - Assume we can approximate R as a product of two low-rank (K) matrices.
  - Learning P and Q allow us to reconstruct all of R



Jure Leskovec, Stanford C246: Mining Massive

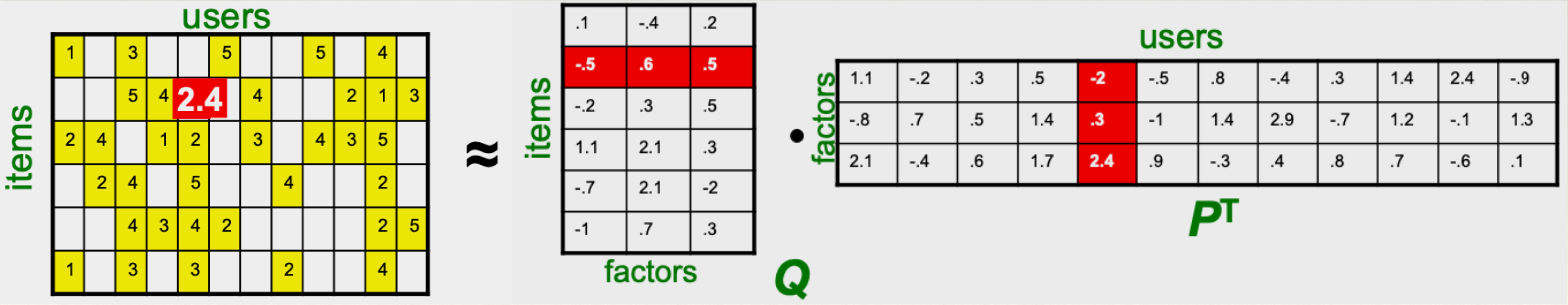
Model based - matrix factorization

- **Motivation**
  - Dimensionality reduction to the same latent space
    - Rows of $P \in \mathbb{R}^{U \times K}$ are low-dim representation of our users.
    - Rows of $Q \in \mathbb{R}^{I \times K}$ are low-dim representation of our items.
  - Compact representation
    - only need to learn and store $UK + IK$ parameters.
  - Matrices can often be adequately represented by low-rank factorization.

Model based - matrix factorization

- Once Q and P are found, estimate the missing values:

$$\hat{r}_{ui} = p_u^T q_i = \sum_{f=1}^{K} p_{uf} q_{if}$$



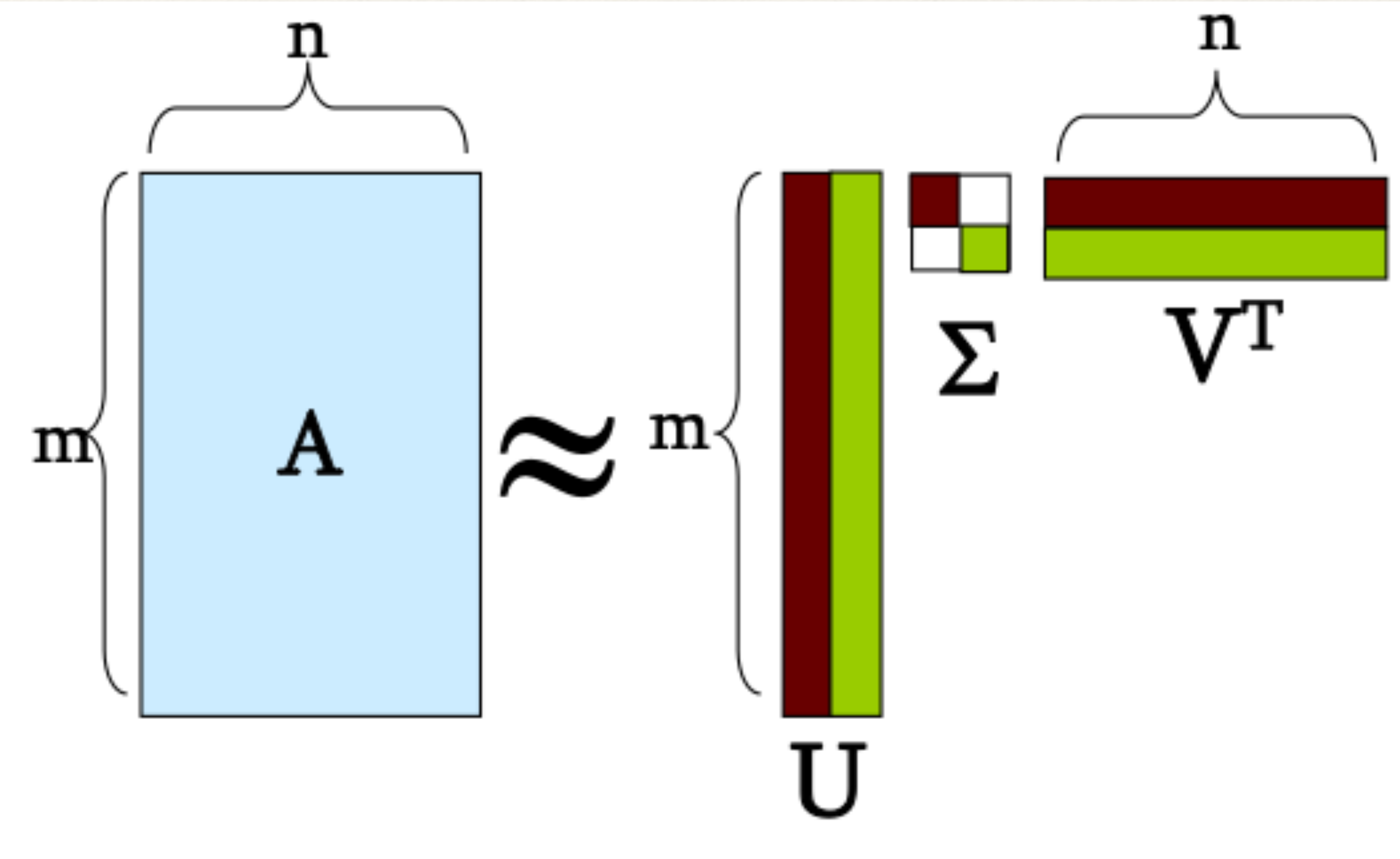
Jure Leskovec, Stanford C246: Mining Massive

Model based - matrix factorization
Singular Value Decomposition

- SVD recap:
    - A - input matrix
    - U - left singular vectors
    - V - right singular vectors
    - Σ - singular values
    - Known to minimise the following:

$$\min_{U,V,\Sigma} \sum_{i,j \in A} \left(A_{ij} - [U \Sigma V^T]_{ij} \right)^2$$

- So, we can write our RMSE error and solve with SVD!
- **Problem:**
    - SVD isn't defined when entries are missing.



Model based - matrix factorization
Non-negative matrix factorization

- Our objective takes into consideration **only the rated** user-item pairs:

$$\min_{P,Q} \sum_{(u,i)\in R} (r_{ui} - p_u^T q_i)^2$$

- In order to prevent overfitting, regularisation term is usually added:

$$\min_{P,Q} \sum_{(u,i)\in R} (r_{ui} - p_u^T q_i)^2 + \frac{\lambda}{2}\left(\|q_i\|^2 + \|p_u\|^2\right)$$

Model based - matrix factorization
Non-negative matrix factorization

- Solving NMF can be done in several ways, some of are:
  - **EM**
  - **Alternating Least-Squares (ALS \ coordinate-decent)**
    - Repeat:
      - Fix P, solve for Q
      - Fix Q, solve for P
  - **Stochastic Gradient Decent**
    - Will not perform well on implicit feedback data
      - Exist in search history, purchased\not, etc.

Model based - matrix factorization
SGD solution

$$\min_{P,Q} \sum_{(u,i)\in R} (r_{ui} - p_u^T q_i)^2 + \frac{\lambda}{2}\left(\|q_i\|^2 + \|p_u\|^2\right)$$

- Initialize P and Q
  - Using SVD, treat missing values as 0.
- Repeat until convergence
  - For each rated entry in R:
    - compute "error" $\epsilon_{ui} = 2(r_{ui} - p_u^T q_i)$
    - $q_i \leftarrow q_i + \alpha_1 \left(\epsilon_{ui} p_u^T - \lambda q_i\right)$
    - $p_u \leftarrow p_u + \alpha_2 \left(\epsilon_{ui} q_i - \lambda p_u^T\right)$

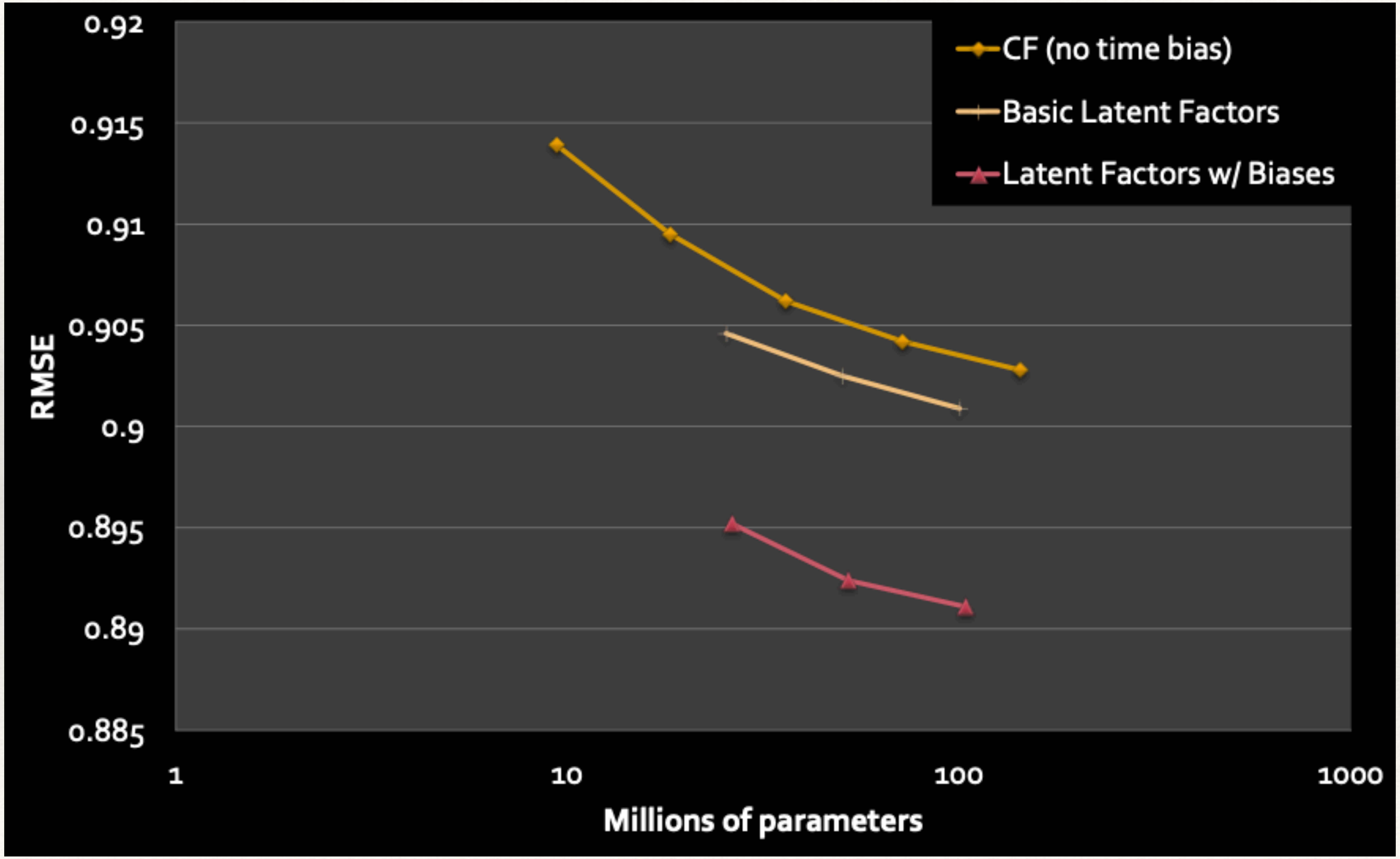
Model based - matrix factorization
Baseline predictor

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

- Adding the *baseline predictor* to the objective function
  - Proved to catch much of the observed signal
  - These user-bias and item-bias are now part of our **learned** parameters.

$$\min_{P,Q,b_*} \sum_{(u,i)\in R} \left(r_{ui} - (\mu + b_u + b_i + p_u^T q_i)\right)^2 + \frac{\lambda}{2}\left(\|q_i\|^2 + \|p_u\|^2 + \|b_u\|^2 + \|b_i\|^2\right)$$

Model based - matrix factorization



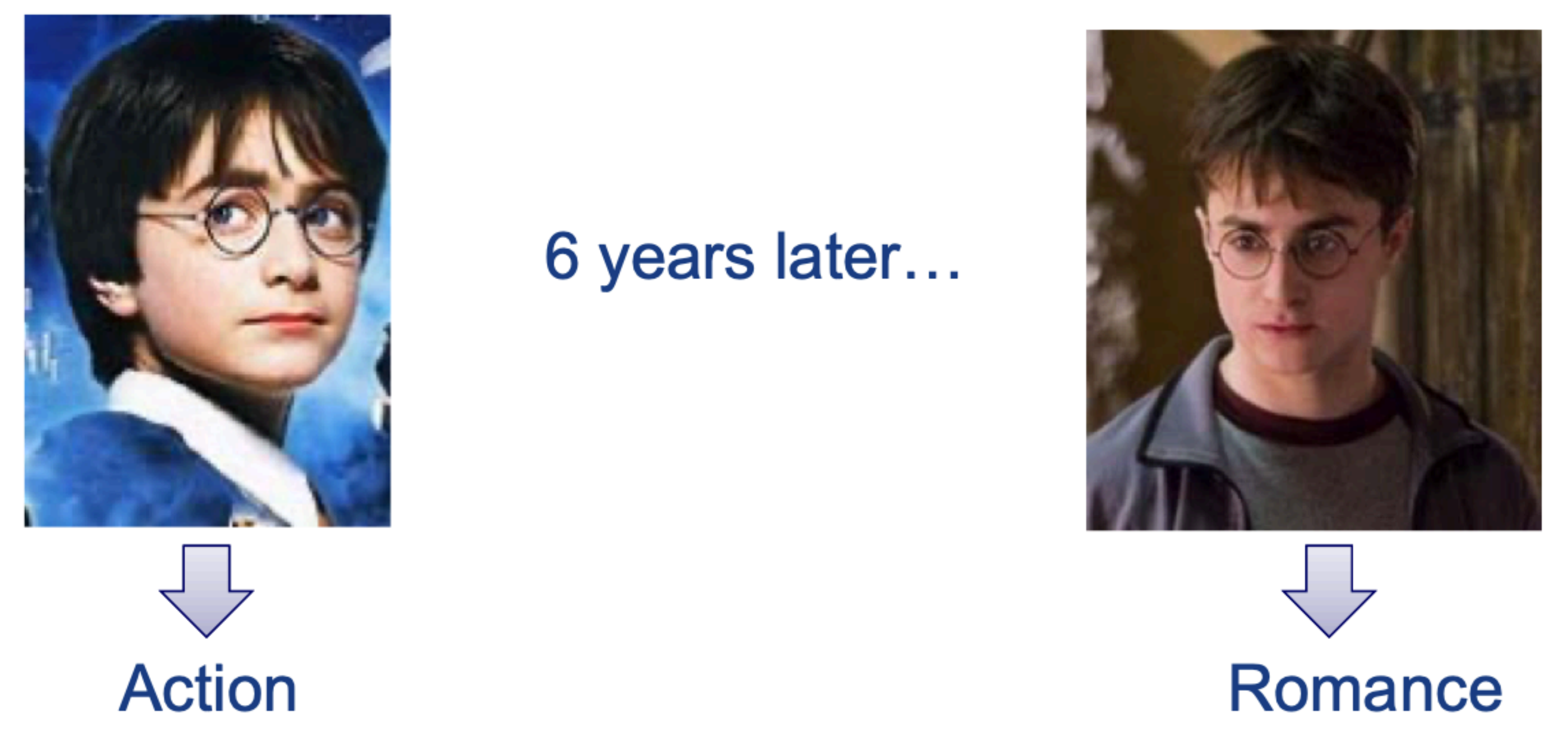
Model based
Improvements - SVD++

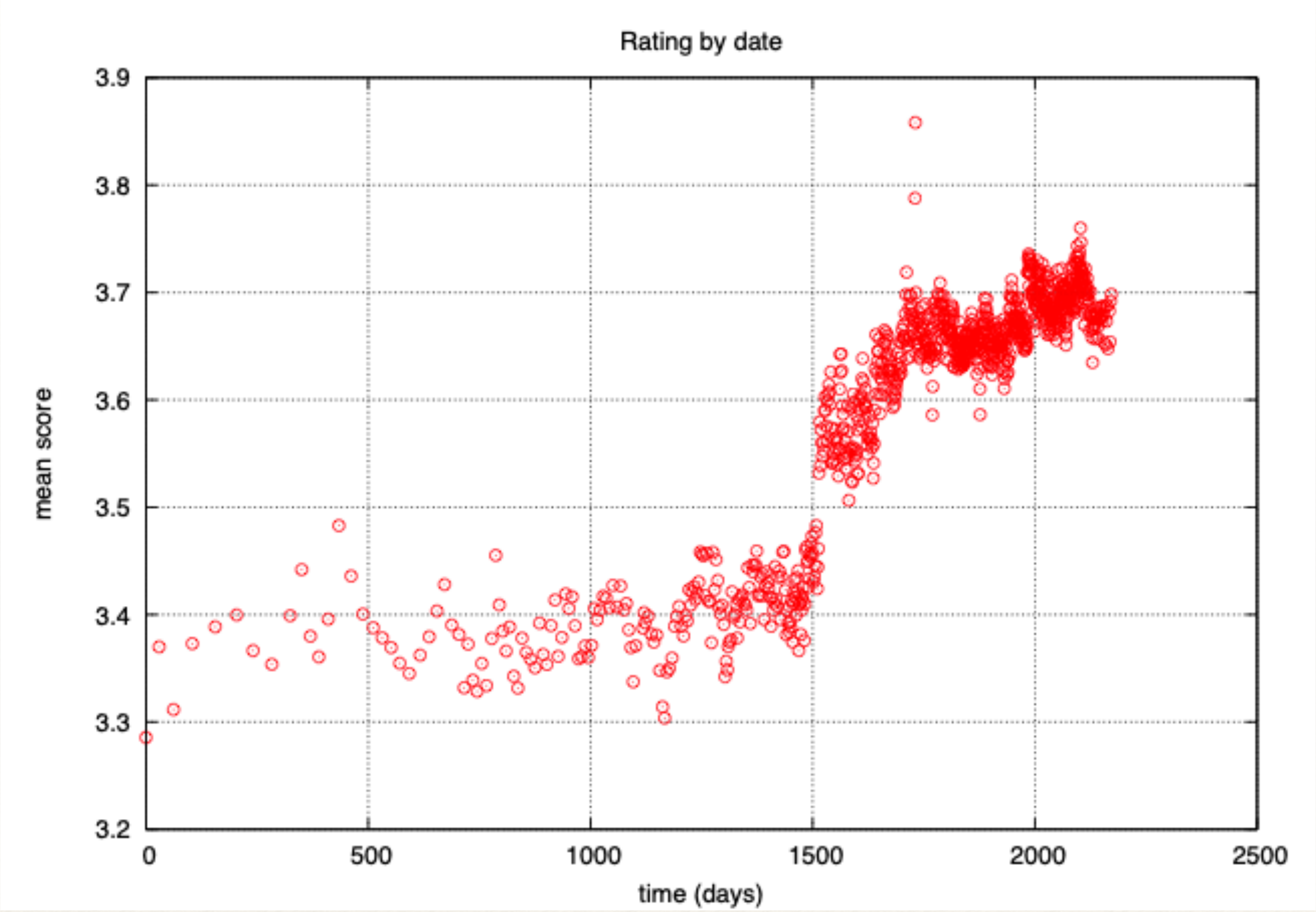
- Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model (Koren, 2008)
    - Characterise users also by the **set of items they rated**, regardless of the rating itself.
        - Incorporates implicit feedback.
        - Regularization is usually added

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} y_j \right)$$

Model based
Improvements - time dependency

- Preferences are time-dependent
  - Season, day of week.
  - User taste evolves.
  - UI improvements.

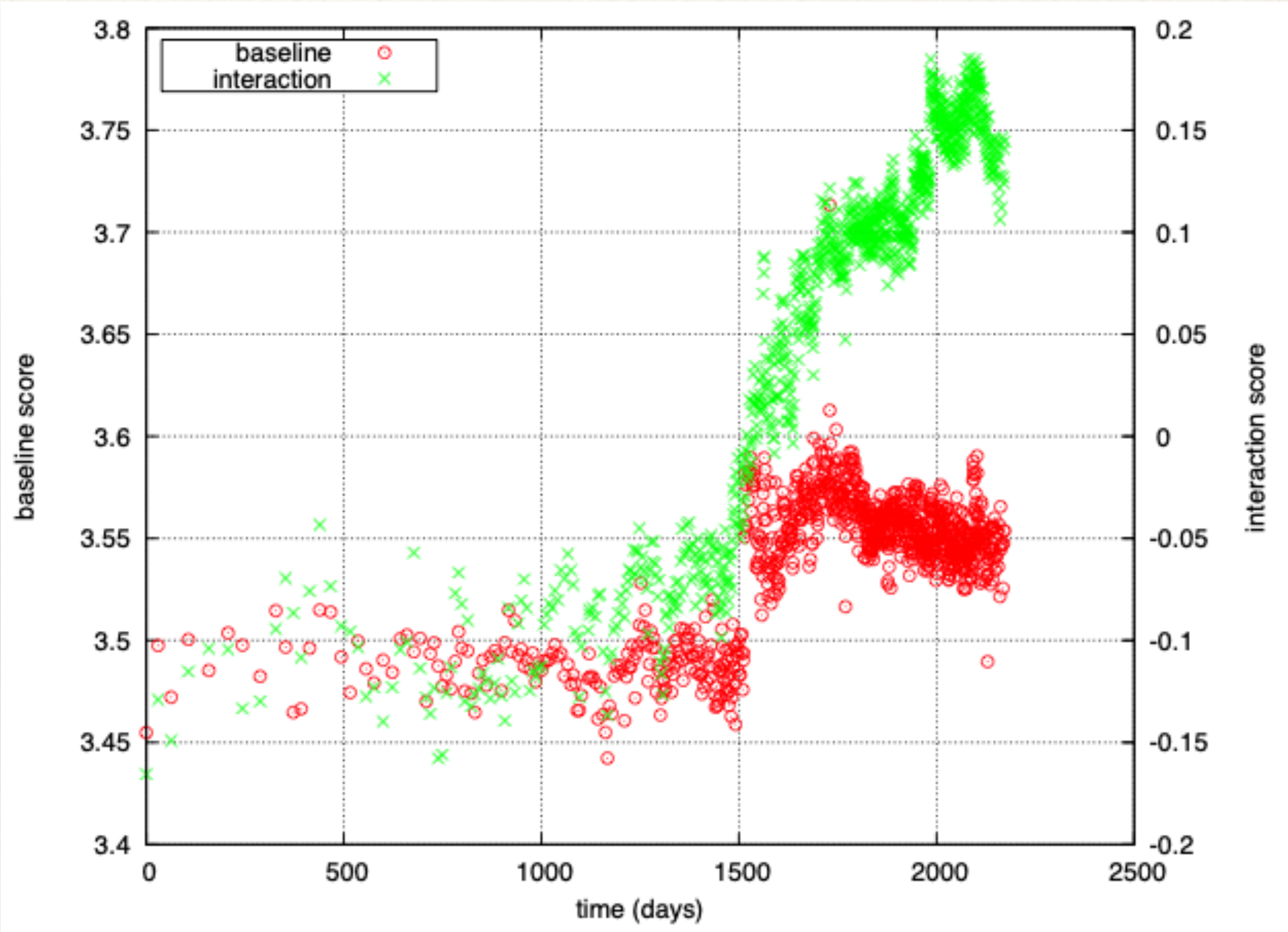




https://pdfs.semanticscholar.org/cf6f/83fcce274606bf0264c59d1c78a30c9c9d18.pdf

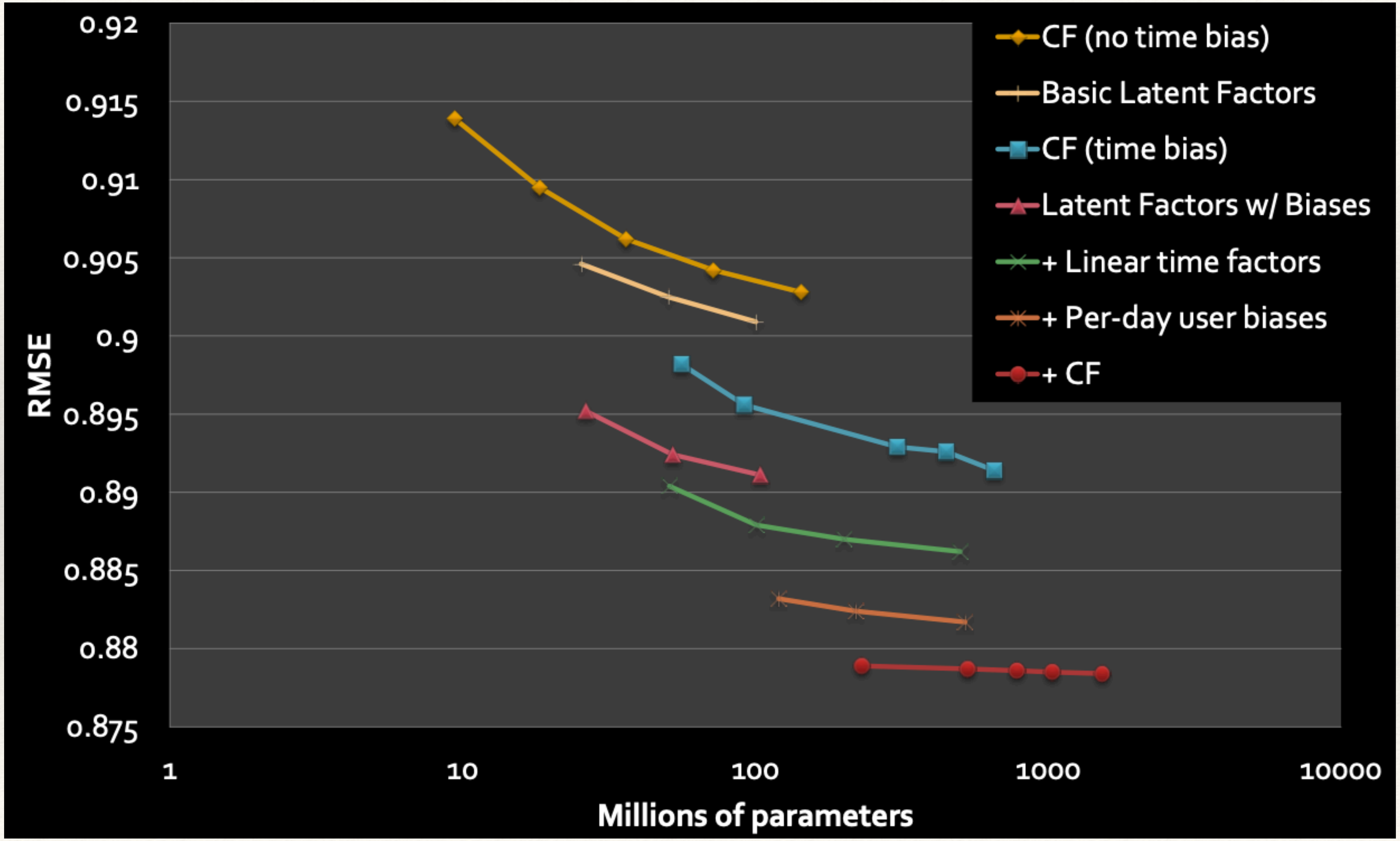
Model based
Improvements - time dependency

- [Collaborative Filtering with Temporal Dynamics](#) (Koren, 2009)
  - $b_{ui}(t) = \mu + b_u(t) + b_i(t)$



$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} y_j \right)$$

Model based
Improvements - time dependency

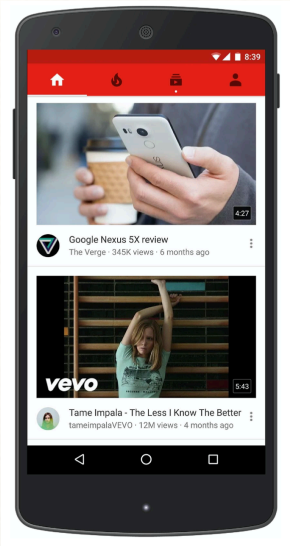


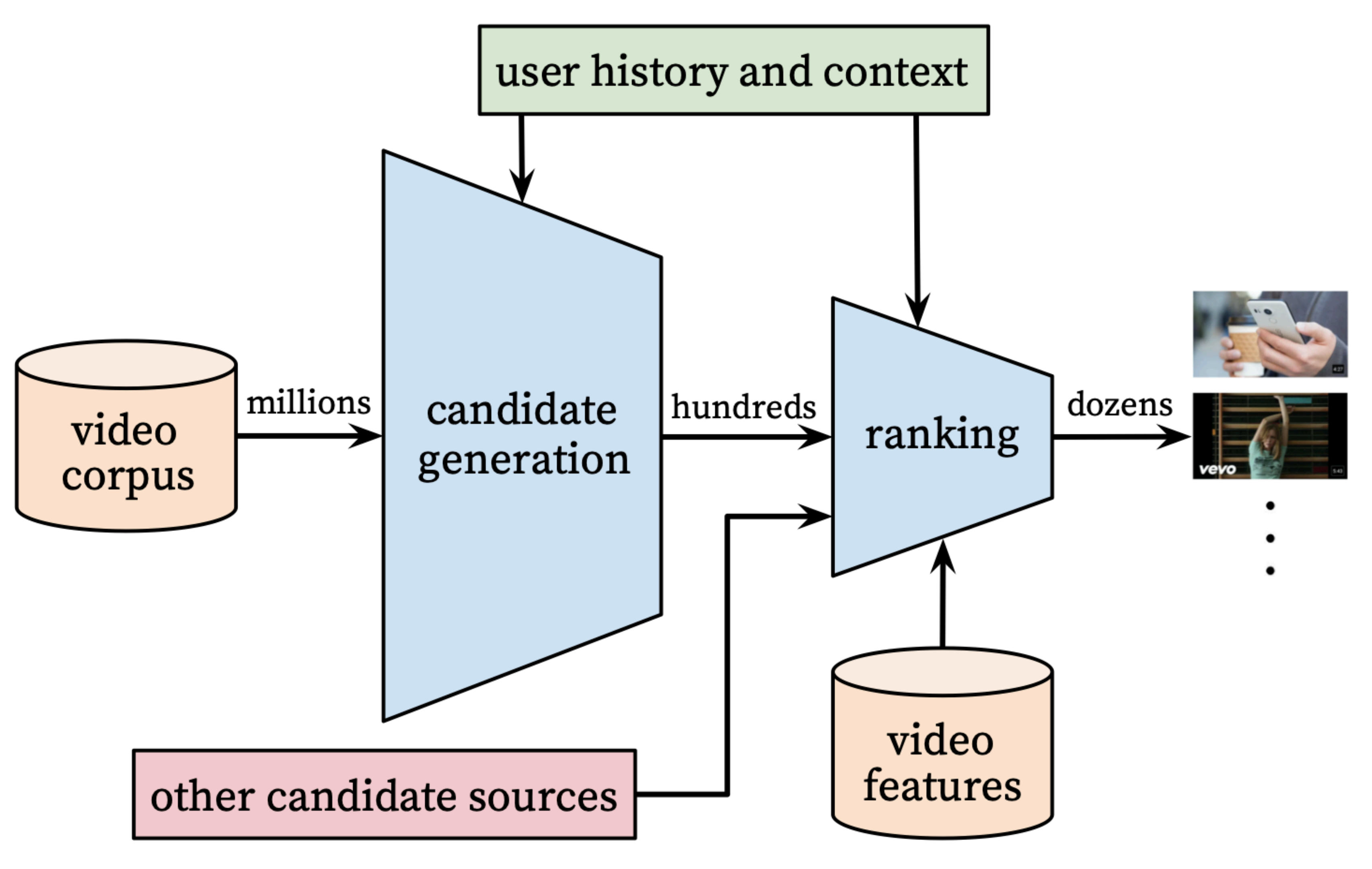
Collaborative filtering
Summary

- CF works and worked well for the past 2-decades.
- Further reading:
  - CofiRank - “Maximum Margin Matrix Factorization for Collaborative Ranking” (Weimer et al.)
  - X. Su and T. M. Khoshgoftaar: A survey of collaborative filtering techniques
- In recent years, some new approach appeared…

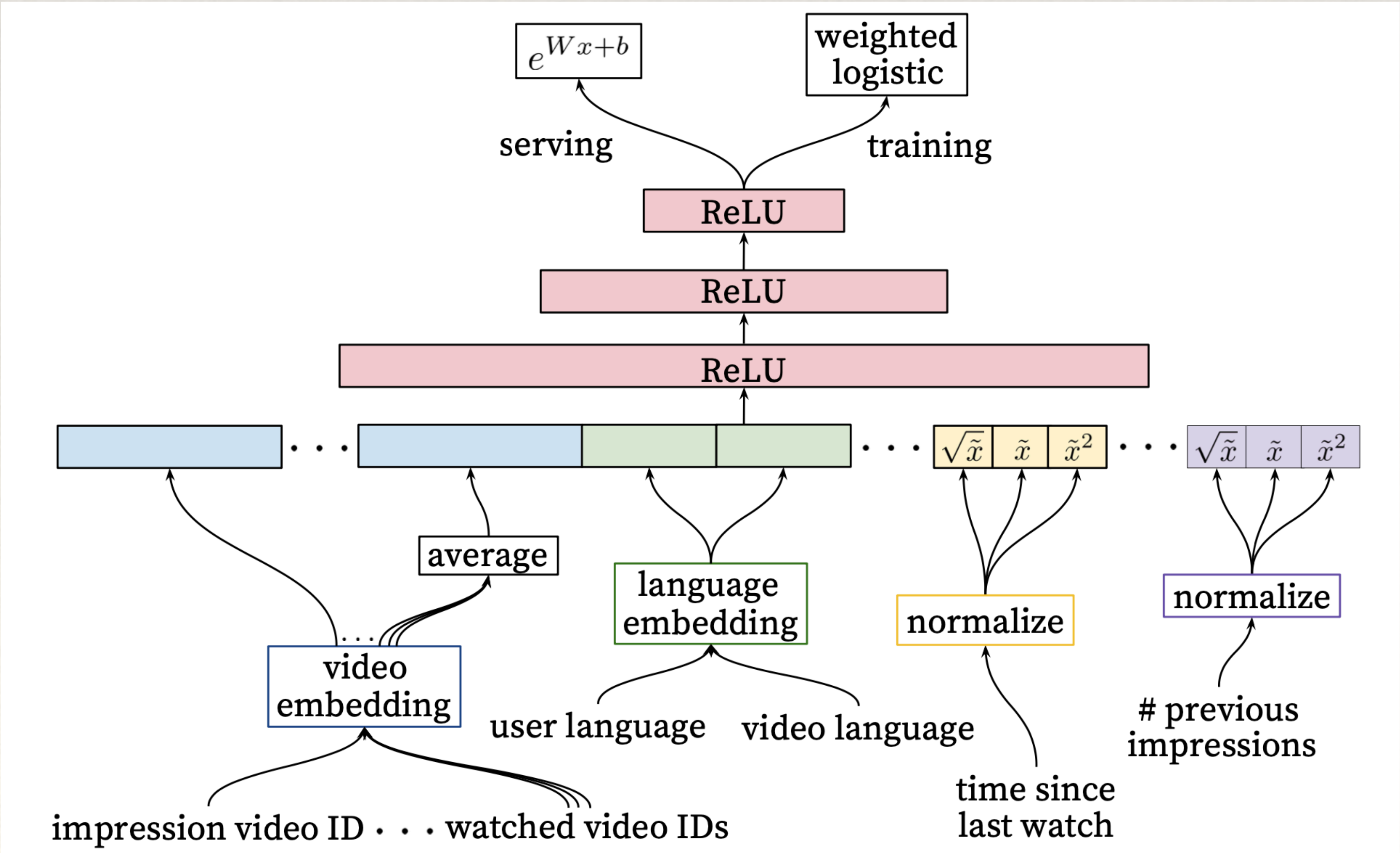
Recent approaches
Neural Recommender Systems

- "Deep Neural Networks for YouTube recommendation" (Covington et al. Google, 2016)



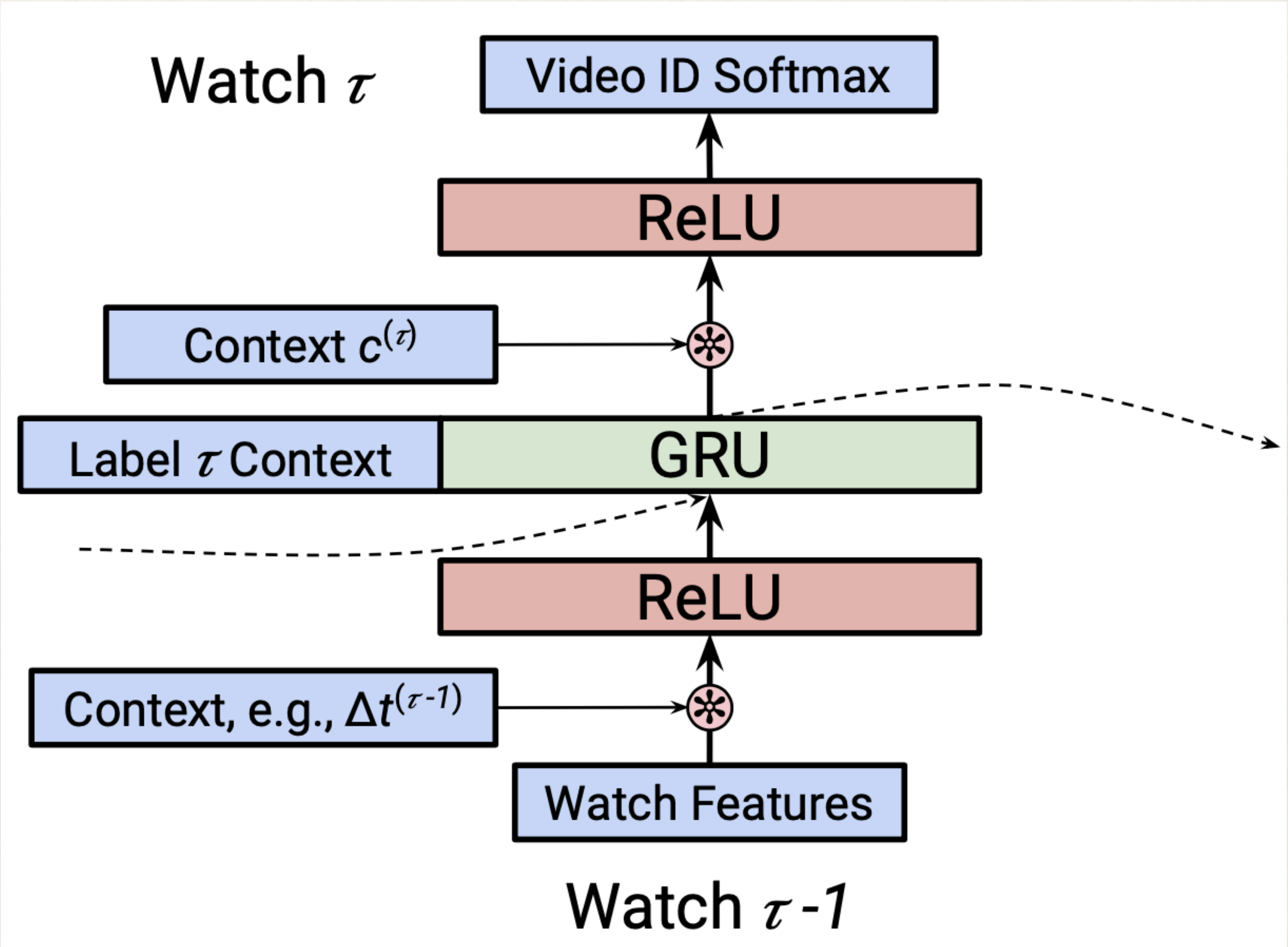


- 2 stages NN
  - Candidate generation - watched videos, search history, video age, user demographics etc.
  - Ranking Network - Video id, previous impression, user-item interactions, search queries



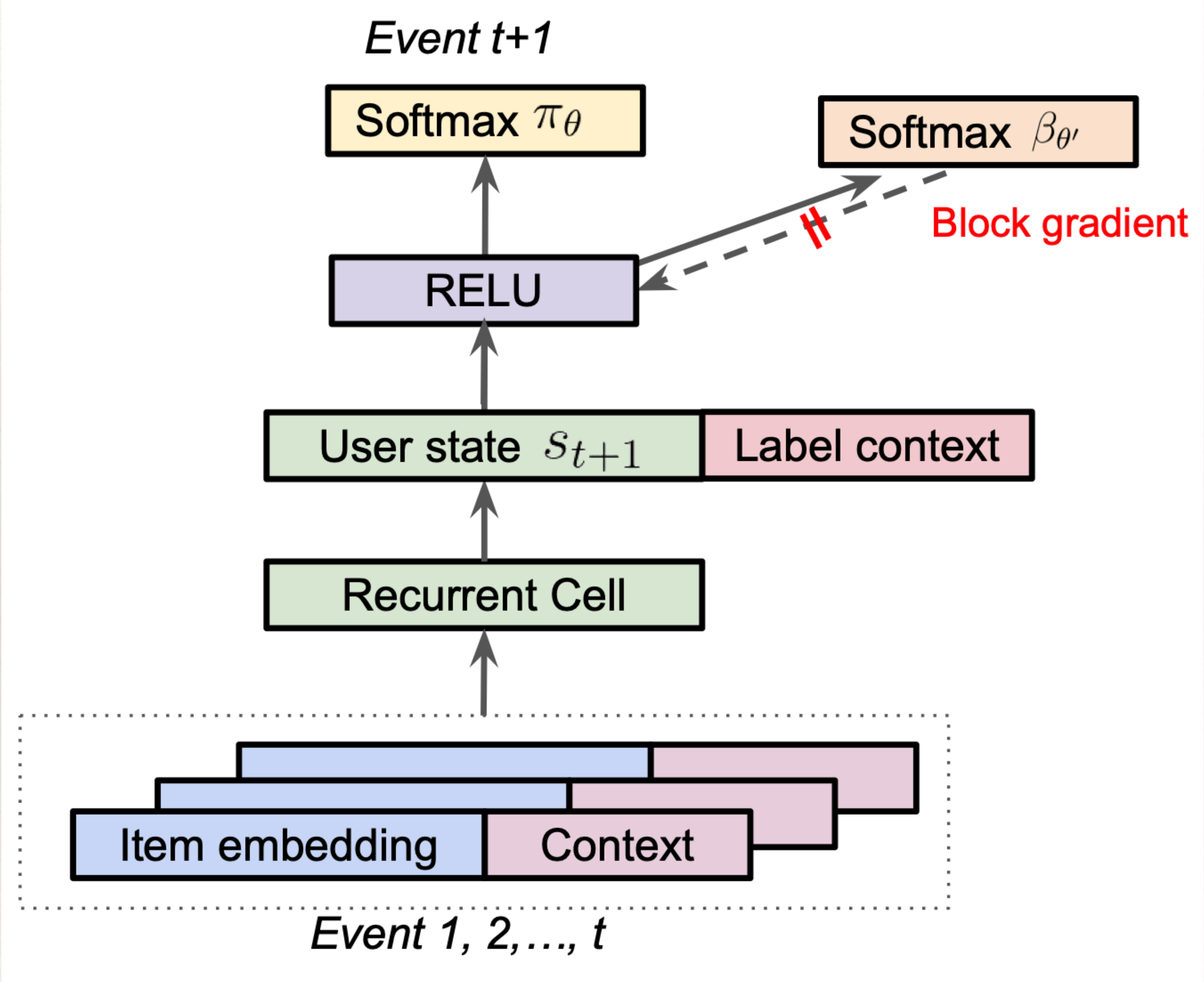
Recent approaches
Neural Recommender Systems

- “Latent Cross: Making Use of Context in Recurrent Recommender Systems” (Beutel et al. Google, 2018)
- RNN-based model that incorporates the contextual features more expressively.
    - Element-wise product of context embedding with the hidden model state, rather than using it as another feature.
    - Kind of mask\attention over the hidden state.



Recent approaches
Neural Recommender Systems

- "Top-K Off-Policy Correction for a REINFORCE Recommender System" (Chen et al., Google 2018)
  - Combining neural recommendation for candidate generation and policy-gradient-based RL approach.



Thank You.

Hope you enjoyed (and learnt)!

segevarbiv@gmail.com