Recommender System

- What is it?
- How to build it?
- Challenges, new directions and state-of-the-art
- R package: **`recommenderlab`**

Recommender System

- **What is it?**
- How to build it?
- Challenges, new directions and state-of-the-art
- R package: `recommenderlab`

A **recommender system** or a **recommendation system** (sometimes replacing "**system**" with a synonym such as platform or engine) is a subclass of information filtering **system** that seeks to predict the "rating" or "preference" a user would give to an item.

Recommender system - Wikipedia
https://en.wikipedia.org/wiki/Recommender_system

- RS is everywhere: Amazon, Wayfair, Netflix, Google News, Pinterest, Spotify, Facebook, Linkedin, OkCupid ……
- A system that can automatically recommend items to users, which are likely to be of interest to the users, by utilizing historical information.

Recommender System

- What is it?
- **How to build it?**
- **Challenges, new directions and state-of-the-art**
- R package: `recommenderlab`

Non-personalized RS

Best Selling books

Top Cyber Monday Deals

Most Popular in Electronics

Best Liked

Top 5 Essential Winter Boots

Recommender System

- What is it?
- **How to build it?**
- **Challenges, new directions and state-of-the-art**
- R package: **`recommenderlab`**

Two Types of Information

1. Characteristic information about the items
2. User-item interactions

Non-personalized RS

Personalized RS

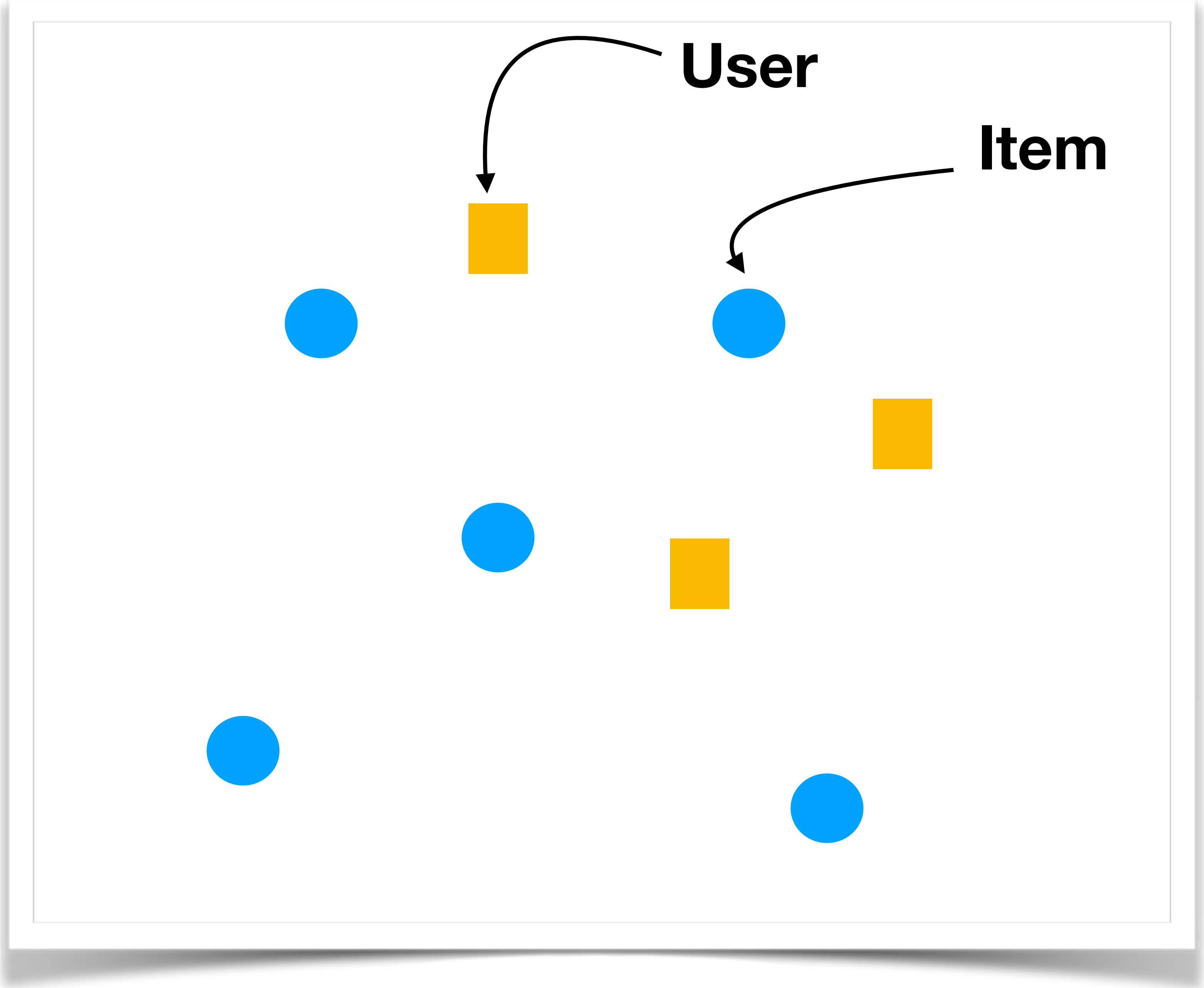
- **Content-based** method
- **Collaborative Filtering** method

  Item-baed CF

  User-baed CF

- **Latent Factor** method
- Hybrid
- **Deep** Recommender System

Content-Based Method



- Item profile: represent each item by a d-dim feature vector. For example, how to characterize a movie/article/product by a feature vector?
- User profile: represent each user by a d-dim feature vector by aggregating the feature vectors of items this user like.

So we embed the m users and n items in a Euclidean space $\mathbb{R}^d$. Then we can recommend items that are close to user i to user i.

Pros

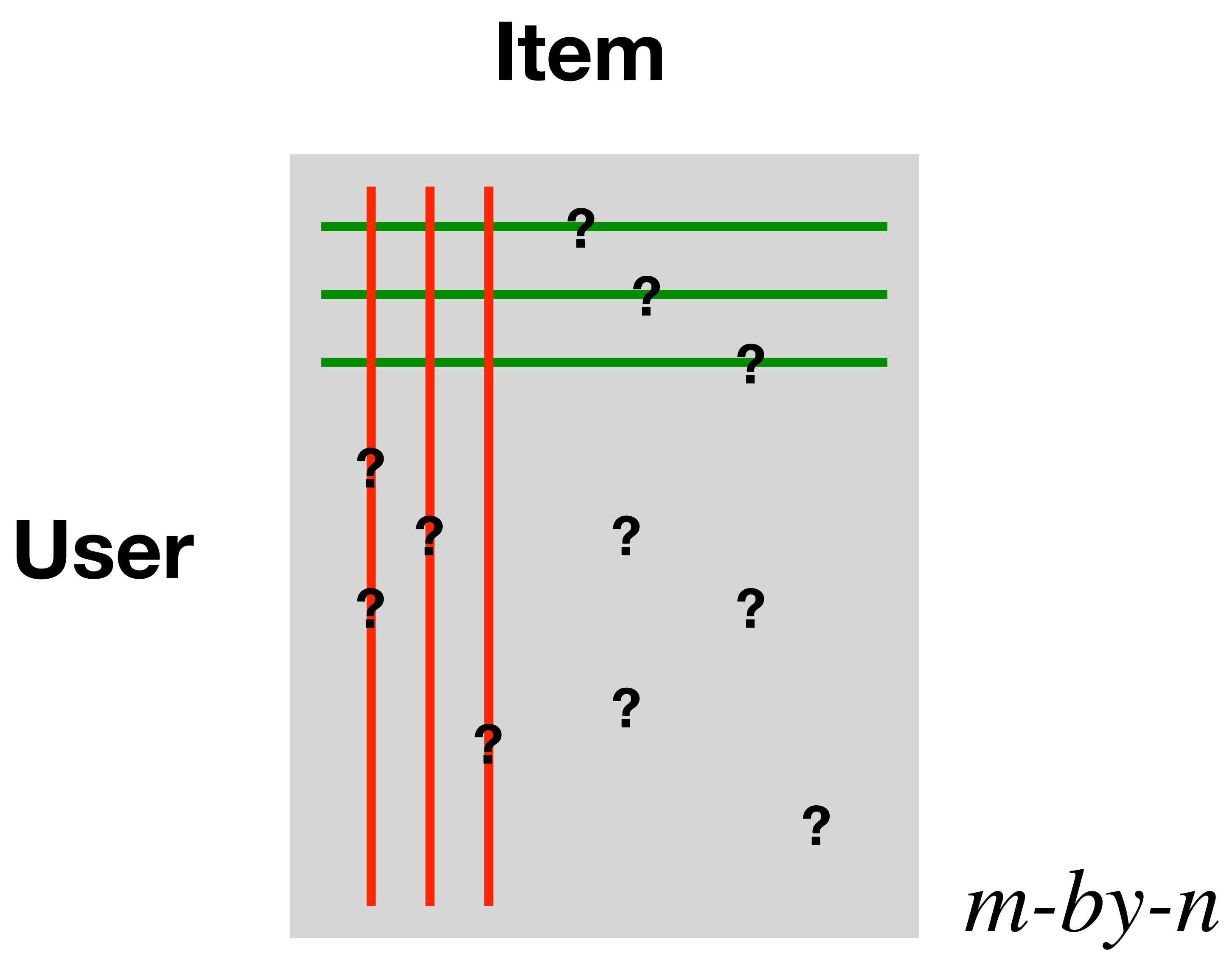
- Do not use user data, so can start recommending on day 1;
- Can recommend new and unpopular items;
- Can recommend to users with unique taste
- Easier to interpret/understand (why we recommend this item to this user)

Cons

- Cannot recommend outside the user's profile
- Recommend substitute not compliment
- **Finding appropriate features is difficult**

Collaborative Filtering (CF) Method

User-Item Rating Matrix: R



How to construct the R matrix?

- Explicit
- Implicit

Challenge: how to differentiate negative vs missing

Collaborative Filtering (CF) Method

User-Item Rating Matrix: R

Item-based CF

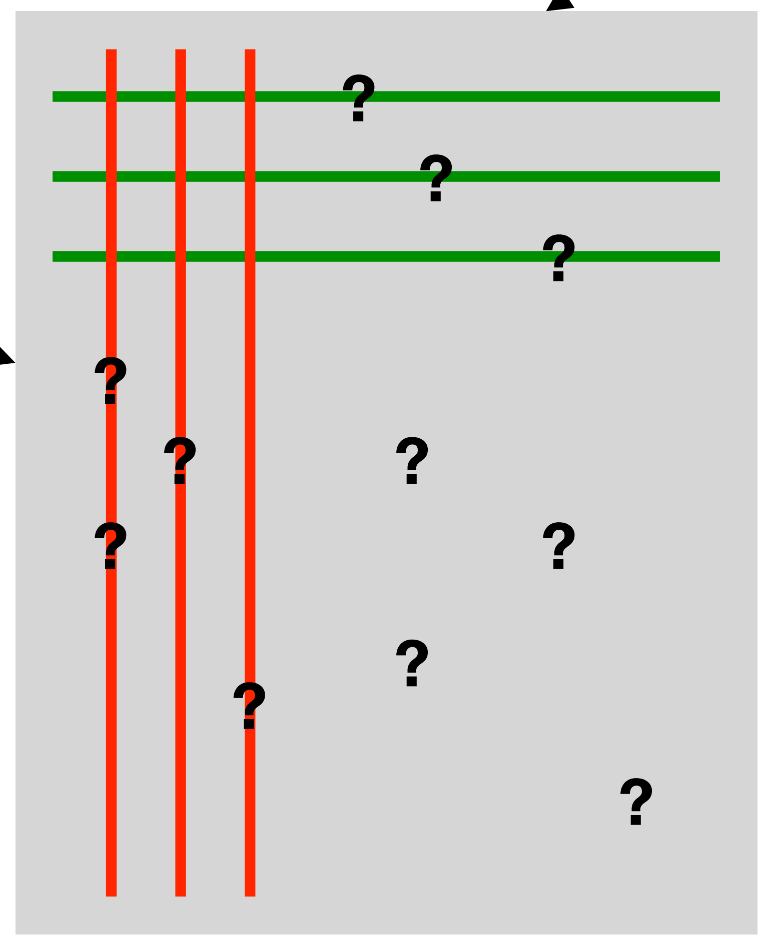
Users tend to like **similar** items

User-based CF

Items tend to be liked by **similar** users

Item

User



m-by-n

	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

Advantage of Centering:

1. Missing = Average instead of zero
2. Handle tough/easy raters

Similarity Measure

- Jaccard similarity: useful for binary ratings

$$\frac{|A \cap B|}{|A| \cdot |B|}, \qquad \text{where } A, B \text{ are two sets.}$$

- Cosine similarity: useful for numerical ratings

$$\frac{u^t v}{\|u\| \cdot \|v\|}, \qquad \text{where } u, v \text{ are two vectors}$$

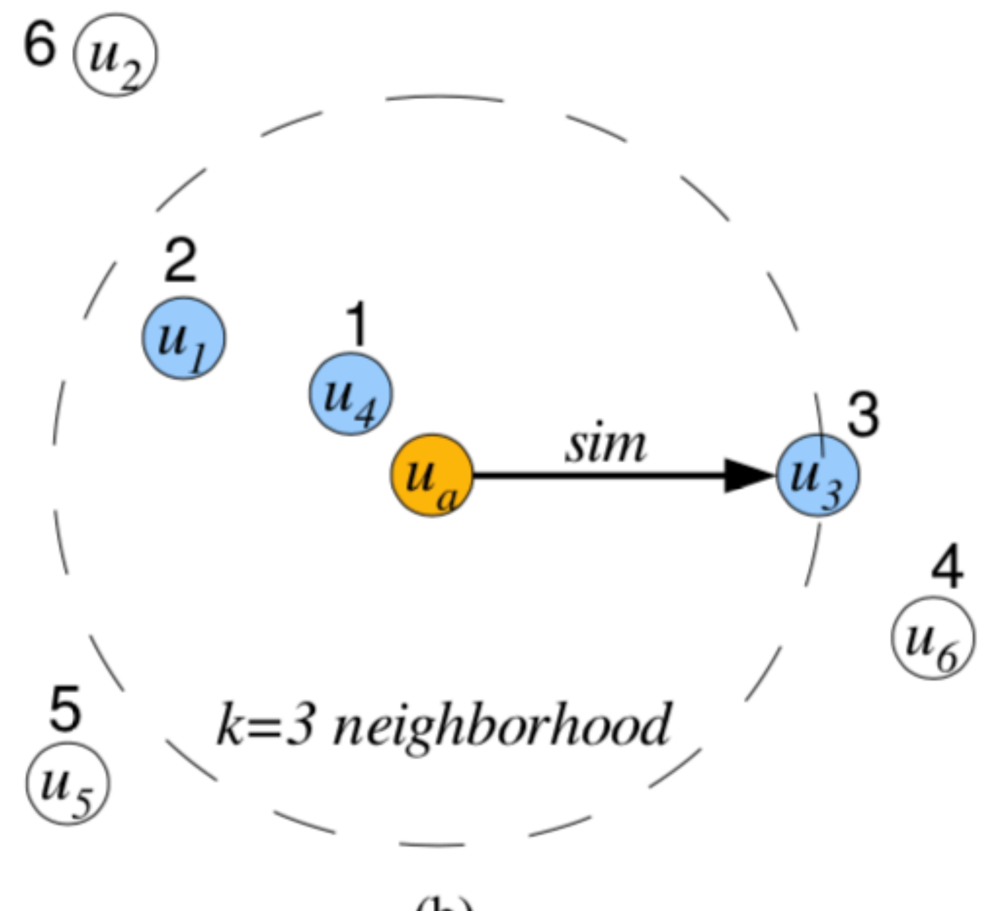
- Centered cosine similarity (Pearson correlation):

$$\frac{(u-\bar{u})^t (v-\bar{v})}{\|u-\bar{u}\| \cdot \|v-\bar{v}\|}, \qquad \text{where } u, v \text{ are two vectors}$$

User-based CF

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

(a)



(b)

Figure 1: User-based collaborative filtering example with (a) rating matrix and estimated ratings for the active user, and (b) user neighborhood formation.

Item-based CF

S	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	$\hat{r}_a$
i_1	-	0.1	0	0.3	0.2	0.4	0	0.1	-
i_2	0.1	-	0.8	0.9	0	0.2	0.1	0	0.0
i_3	0	0.8	-	0	0.4	0.1	0.3	0.5	4.6
i_4	0.3	0.9	0	-	0	0.1	0	0.2	3.2
i_5	0.2	0	0.4	0	-	0.1	0.2	0.1	-
i_6	0.4	0.2	0.1	0.3	0.1	-	0	0.1	2.0
i_7	0	0.1	0.3	0	0.2	0	-	0	4.0
i_8	0.1	0	0.5	0.2	0.1	0.1	0	-	-
u_a	2	?	?	?	4	?	?	5	

k=3

Figure 2: Item-based collaborative filtering

Source: recommenderlab: A Framework …. By Michael Hahsler (File on Piazza)

= (3.0 + 4.0)/2

=(1.0 + 2.0 + 1.0)/3

0.0 = 3NN are missing
4.6 = (0.4/0.9)(4) + (0.5/0.9)(5)
3.2 = (0.3/0.5)(2) + (0.2/0.5)(5)

Note: Neighborhood should vary with items, i.e., choose neighbors who also rated item *i*.

Content-Based

Pros

- Do not use user data, so can start recommending on day 1;
- Can recommend new and unpopular items;
- Can recommend to users with unique taste
- Easier to interpret/understand (why we recommend this item to this user)

Cons

- Cannot recommend outside the user’s profile
- Recommend substitute not compliment
- Finding appropriate features is difficult

Computation Challenge for CF: how to efficiently find kNN in a large data set?

Collaborative Filtering (CF)

Cons

- Need enough user data to start recommendation; cannot operate on day 1
- Cannot recommend new, unrated items
- Tend to recommend popular items, against the purpose of personalized RS
- **Cold start** problem for new users/items

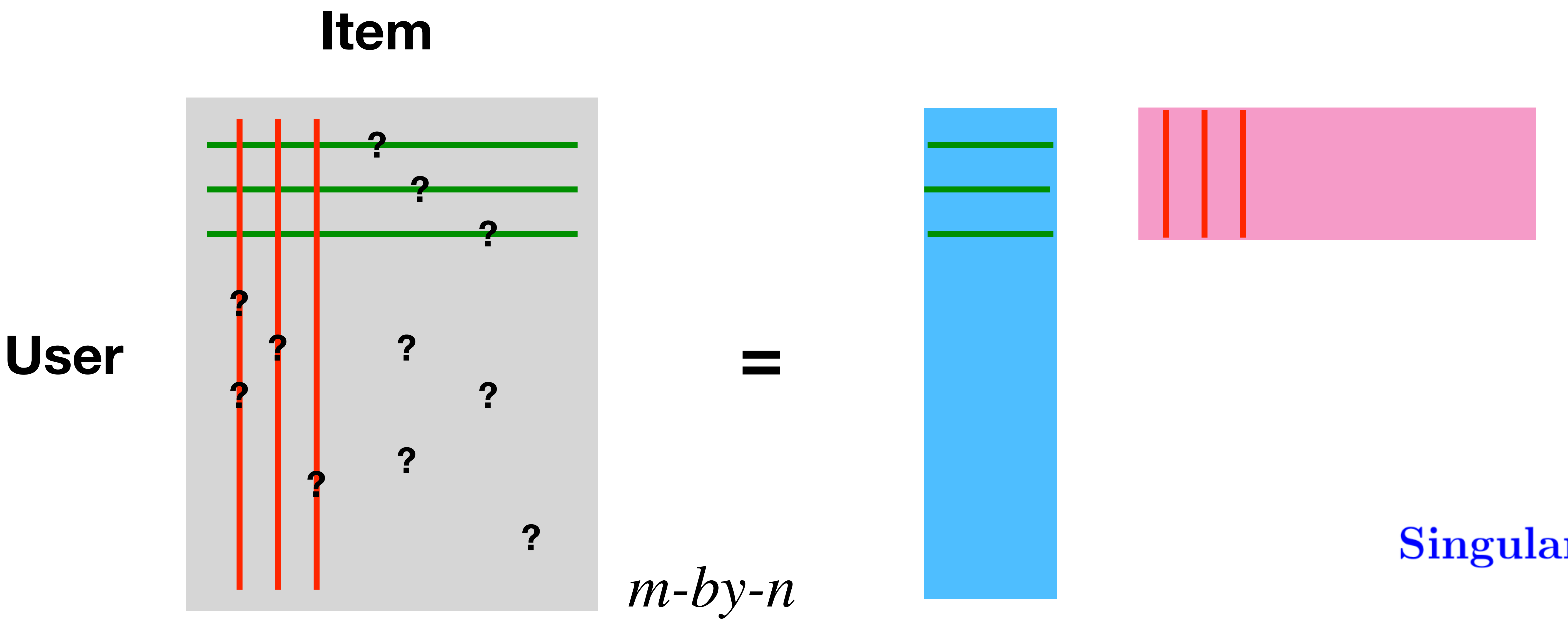
Pros

- No need to define features
- Can recommend outside the user’s profile
- Recommend substitute not compliment

Item-based performs better in practice: easier to find similar items, but difficult to find similar people

Latent Factor Model

User-Item Rating Matrix: R



Singular Value Decomposition

Approximate $R_{m\times n} \approx U_{m\times d} V^t_{d\times n}$ by minimizing

$$\sum_{R_{ij}\neq \mathrm{NA}} (R_{ij} - u_i^t v_j)^2 + \lambda_1 \mathrm{Pen}(U) + \lambda_2 \mathrm{Pen}(V),$$

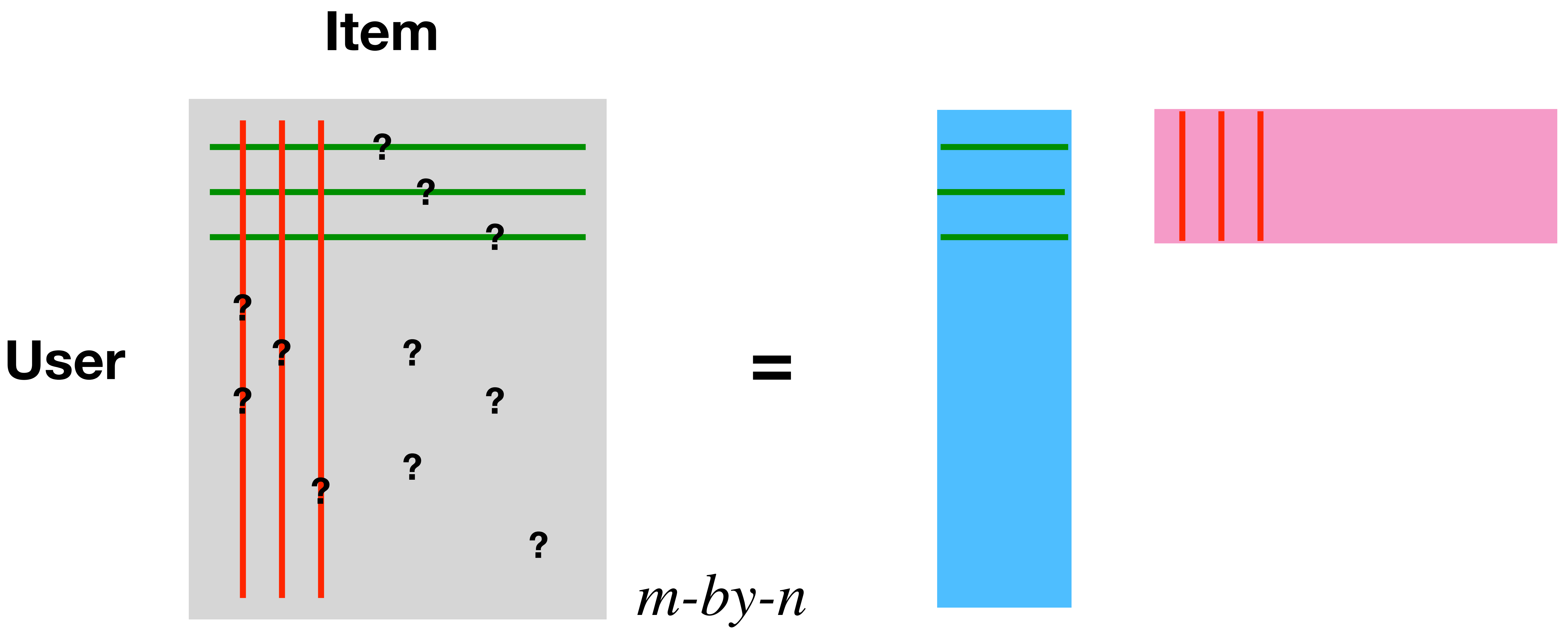
where u_i is the i-th row of matrix U and v_j is the j-th row of matrix V.

Then we can predict any missing entries in R by the corresponding inner product of u_i and v_j.

The classical **SVD** algorithm isn't applicable here due to missing entries, instead algorithms based on **Stochastic Gradient Descent** are employed in practice.

The Global Base Line Model: Correct Bias

User-Item Rating Matrix: R



Over-all Average

Remaining Interaction Term : Collaborative Filtering or Latent Factor model on the Remaining Interaction Term

$$R_{ij} = \mu + a_i + b_j + \tilde{R}_{ij}$$

User effect **Movie effect**

Some Practical Issues

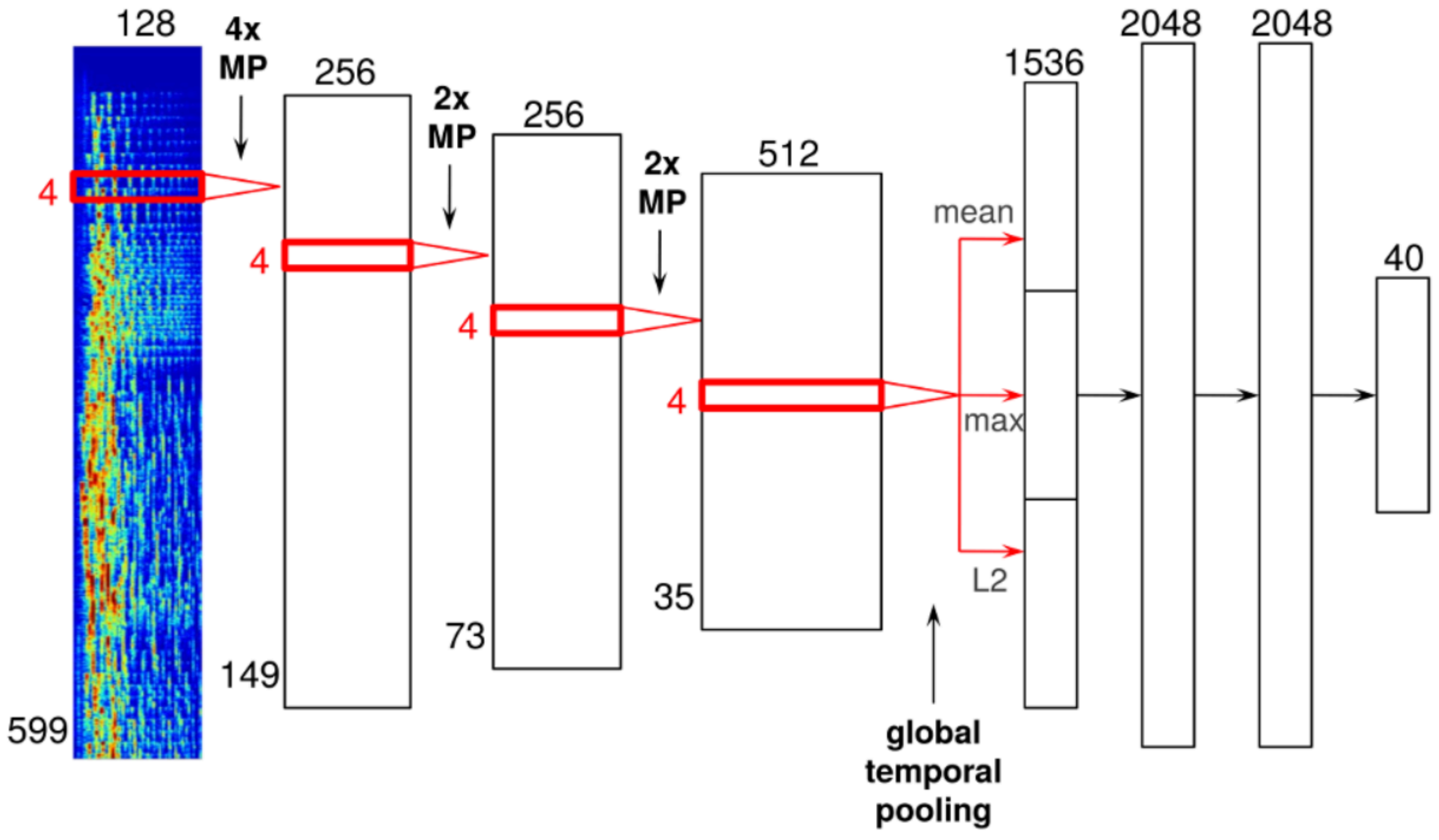
- **Cluster** users and items to reduce computation
- **Hybrid**: combine multiple recommender systems
- Different **contexts** (location, time, device) and **interface** (computer, mobile) need different recommendation systems.
- How to evaluate a recommender system?
  - **RMSE** vs **Top-k**
  - **Serendipity**/**Diversity** versus **Accuracy**
- How to incorporate user **feedback**

Challenges

- **Scalability**: large amount of users and items
- **Sparsity** of the data
- **Utility matrix**: how to construct it based the problem at hand
- **Cold-start**: how to recommend a new item or make recommendation to a new user

Deep Recommender Systems

- Use Deep Learning to construct latent factors for items/users
- Train a Deep Learning model to learn the preference between users and items



http://benanne.github.io/2014/08/05/spotify-cnns.html

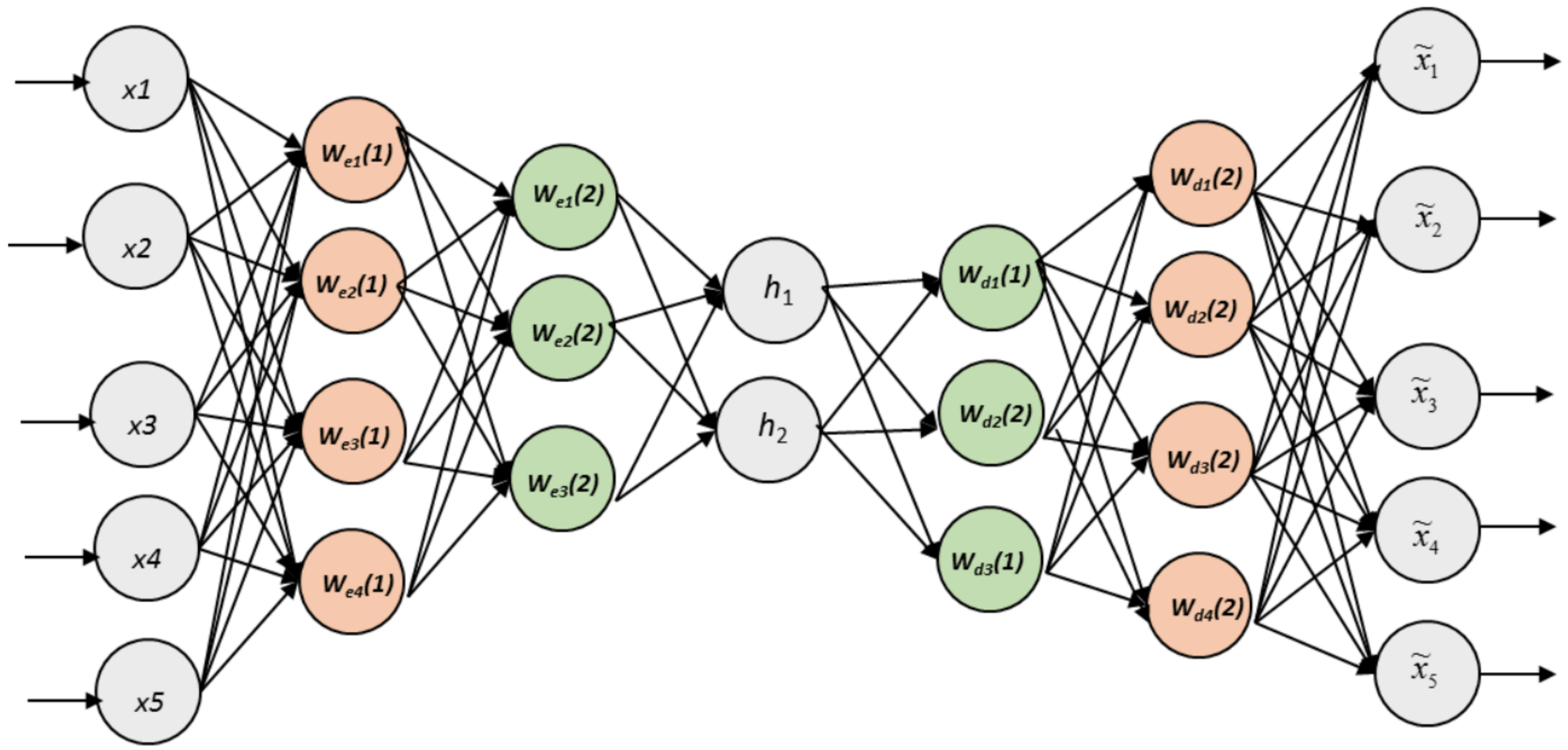


Fig. 2. Deep Autoencoder architecture.

https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d

Deep Recommender Systems

- Use Deep Learning to construct latent factors for items/users
- Train a Deep Learning model to learn the preference between users and items

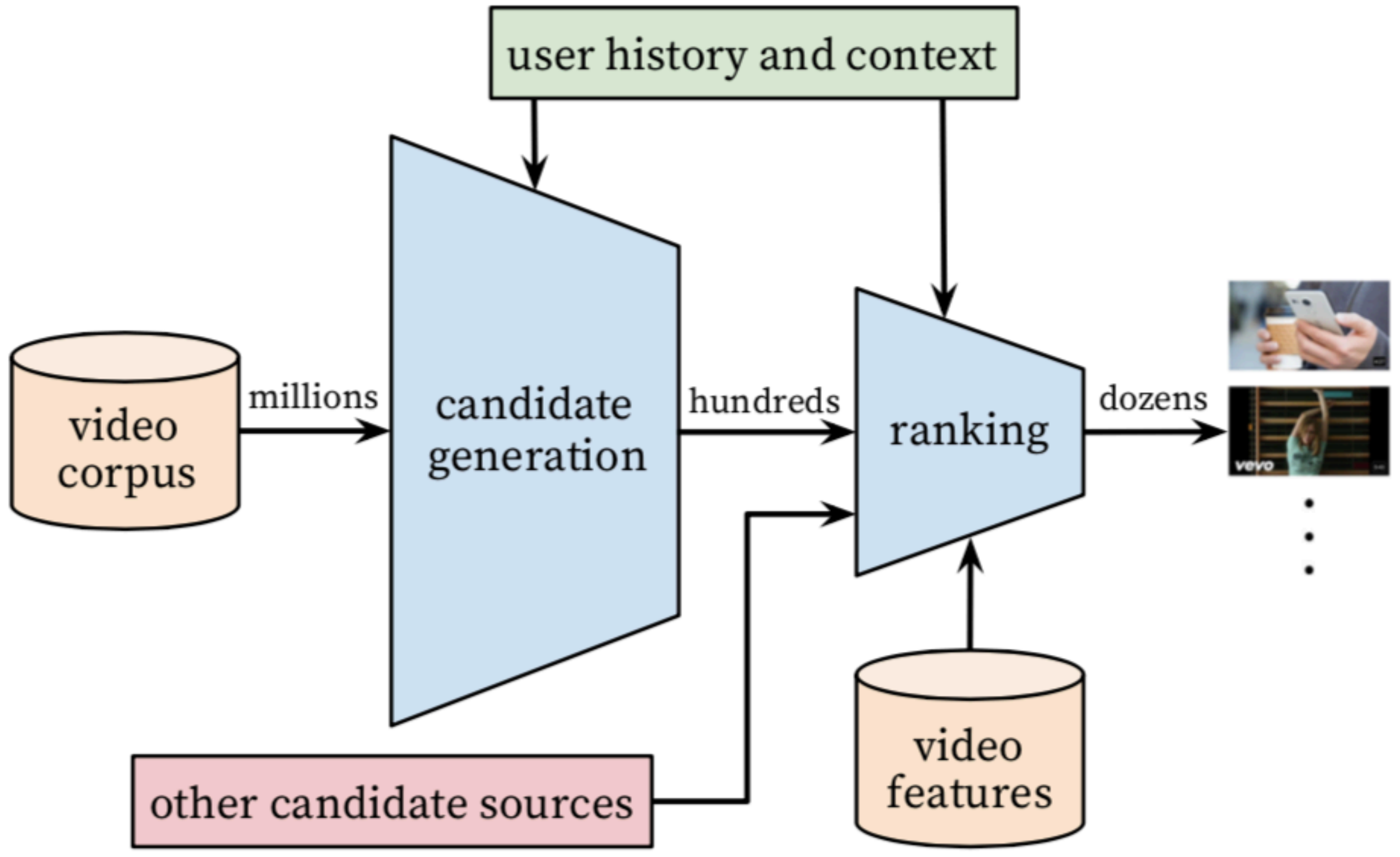
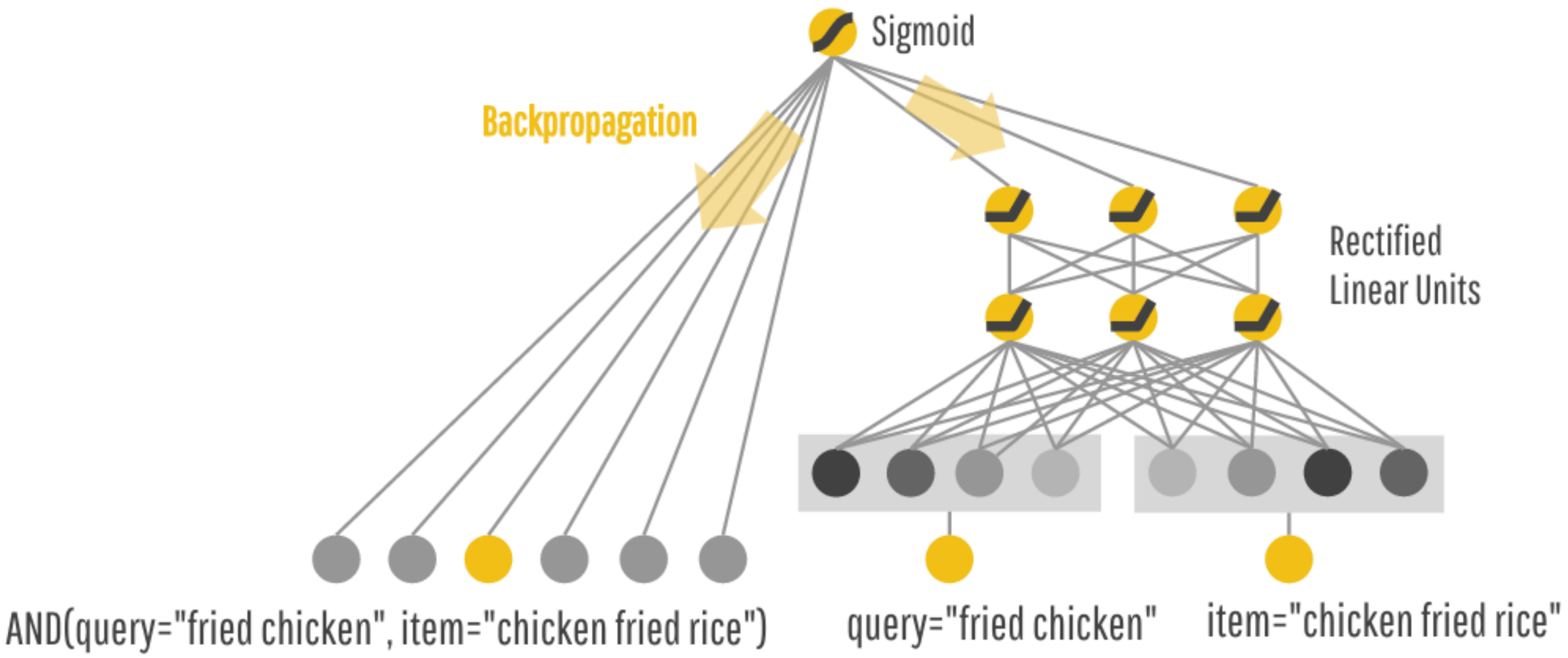
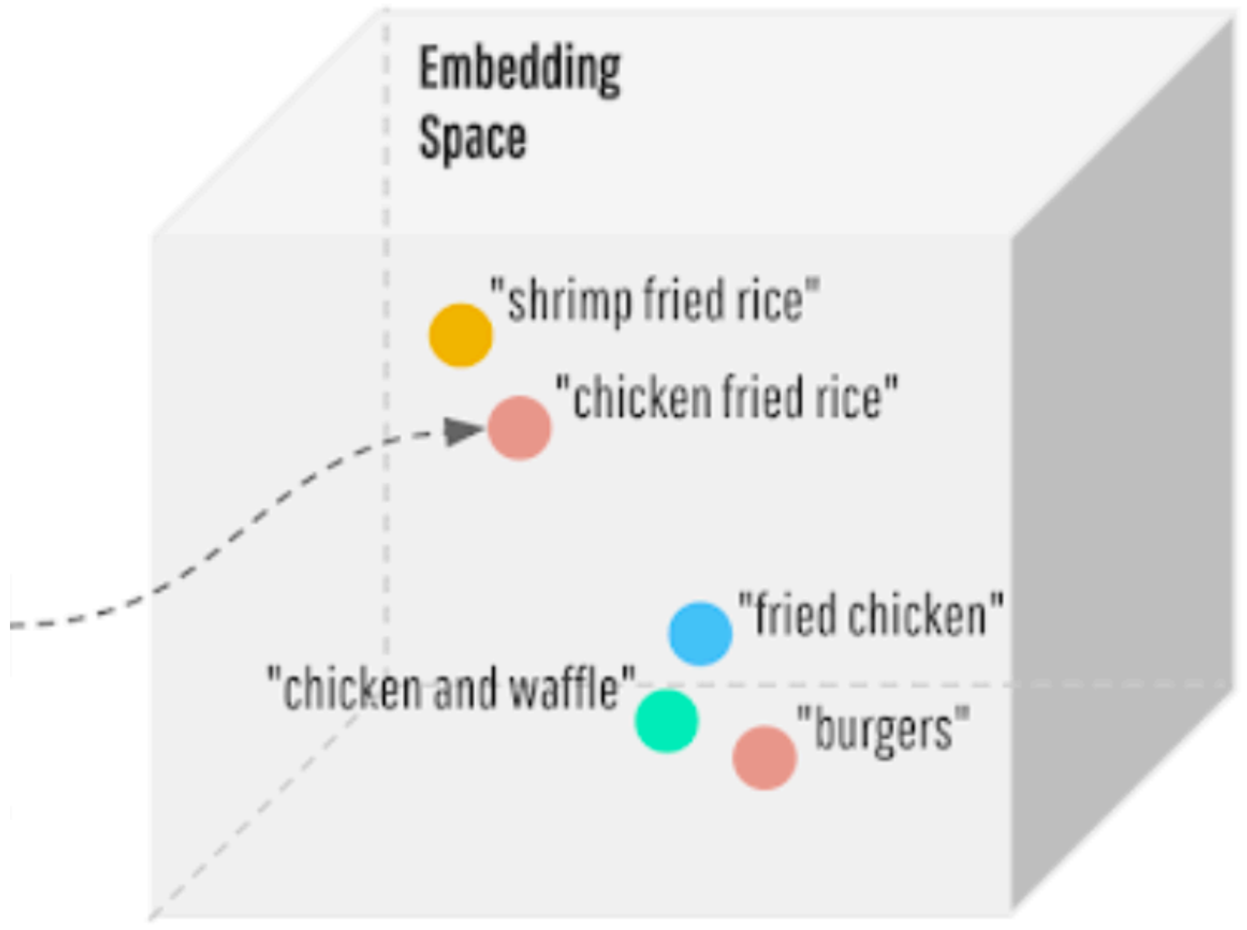


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

Covington et al. (2016)





Google's wide-and-deep model

- Wide (sparse) linear model for **memorization**
- Deep neural network model for **generalization**