COMP4222 Machine Learning with Structured Data

Recommender Systems 1

Instructor: Yangqiu Song

Slides credits: Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich, Max Welling, Dan Jurafsky

Problem domain

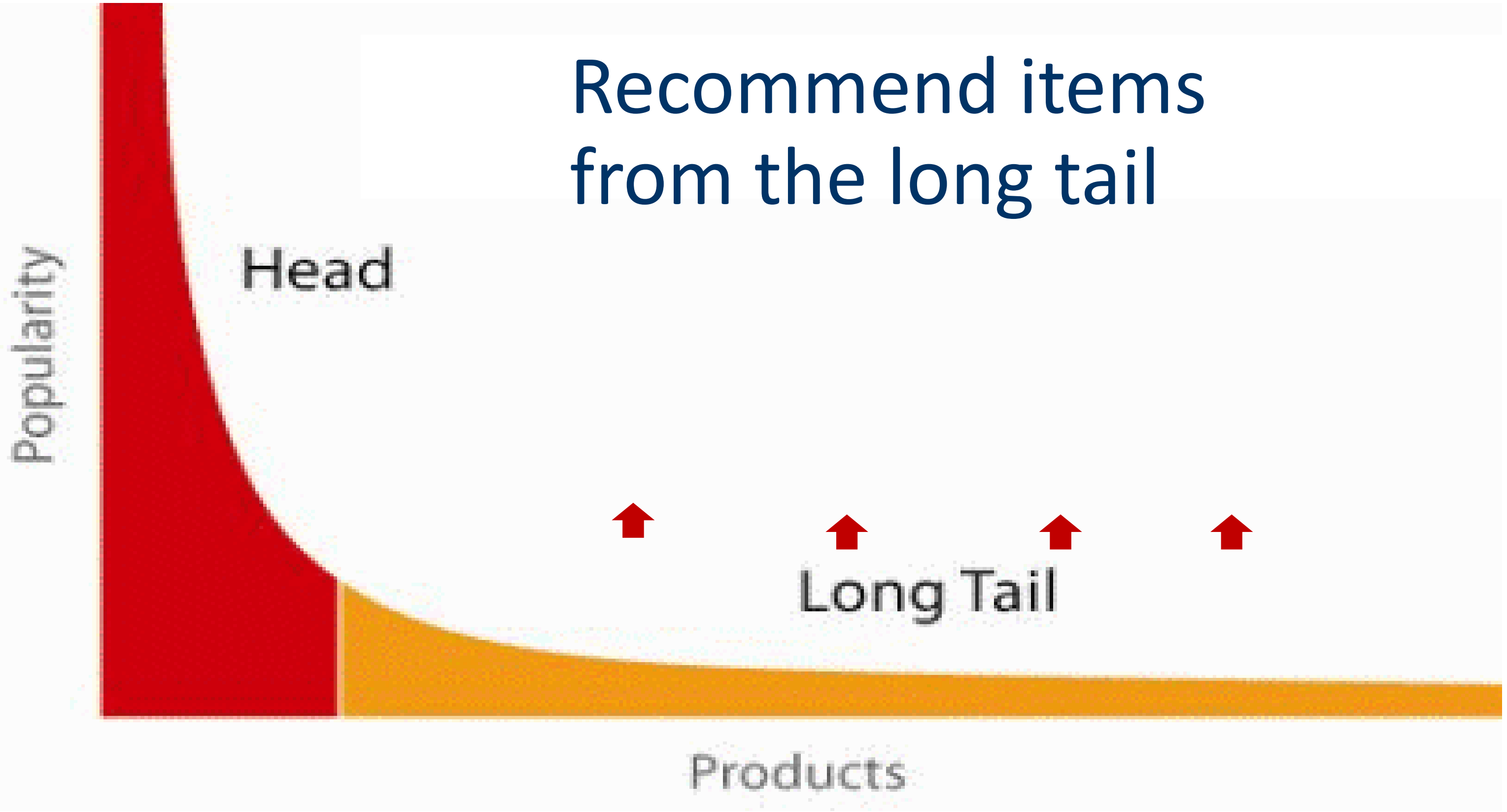
- Recommendation systems (RS) help to match users with items
  - Ease information overload
  - Sales assistance (guidance, advisory, persuasion,…)

RS are software agents that elicit the interests and preferences of individual consumers […] and make recommendations accordingly.

They have the potential to support and improve the quality of the decisions consumers make while searching for and selecting products online.

- (Xiao & Benbasat 2007[1])

When does a RS do its job well?



"Recommend widely unknown items that users might actually like!"

- Items rated > 3 in MovieLens 100K dataset
  - 20% of items accumulate 74% of all positive ratings

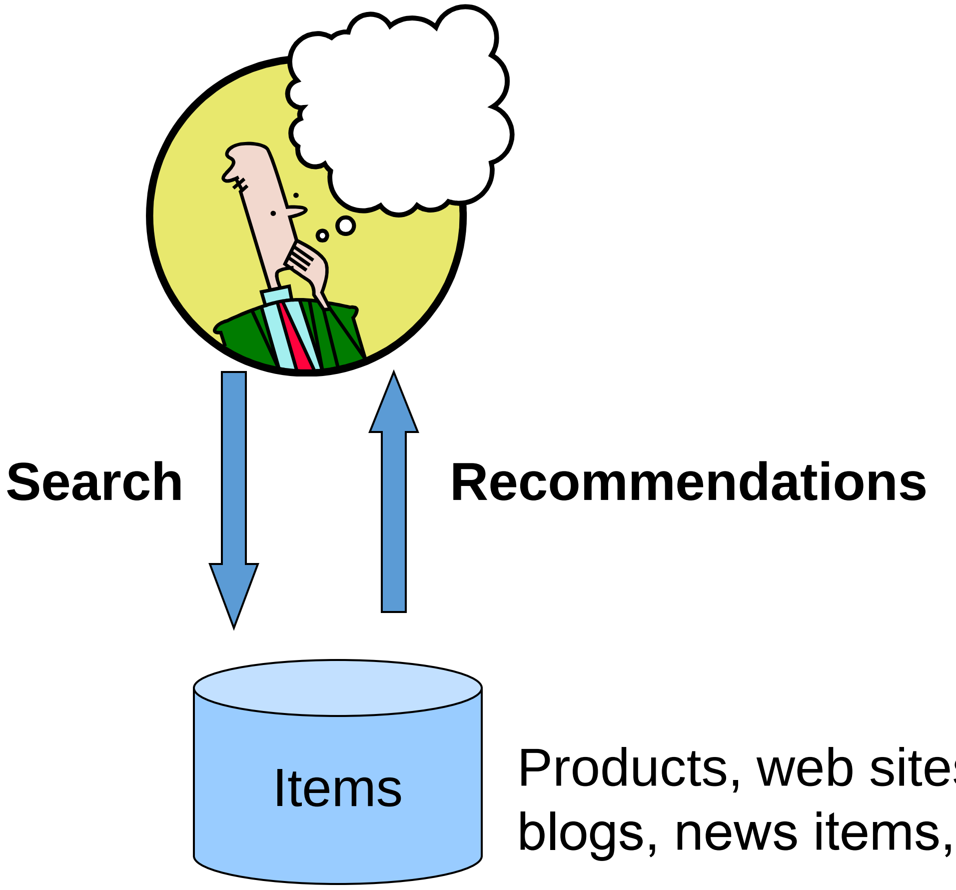
Recommender systems

- RS seen as a function
- Given:
  - User model (e.g. ratings, preferences, demographics, situational context)
  - Items (with or without description of item characteristics)
- Find:
  - Relevance score. Used for ranking.

- Relation to Information Retrieval:
  - IR is finding material [..] of an unstructured nature [..] that satisfies an information need from within large collections [..].

(1) Manning, Raghavan, and Schütze, *Introduction to information retrieval*, Cambridge University Press, 2008

Recommendations



Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

NETFLIX

YouTube

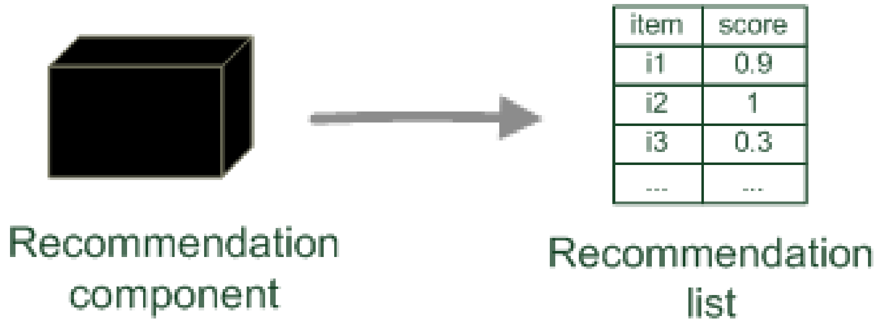
movielens
helping you find the *right* movies

last.fm
the social music revolution

Google News

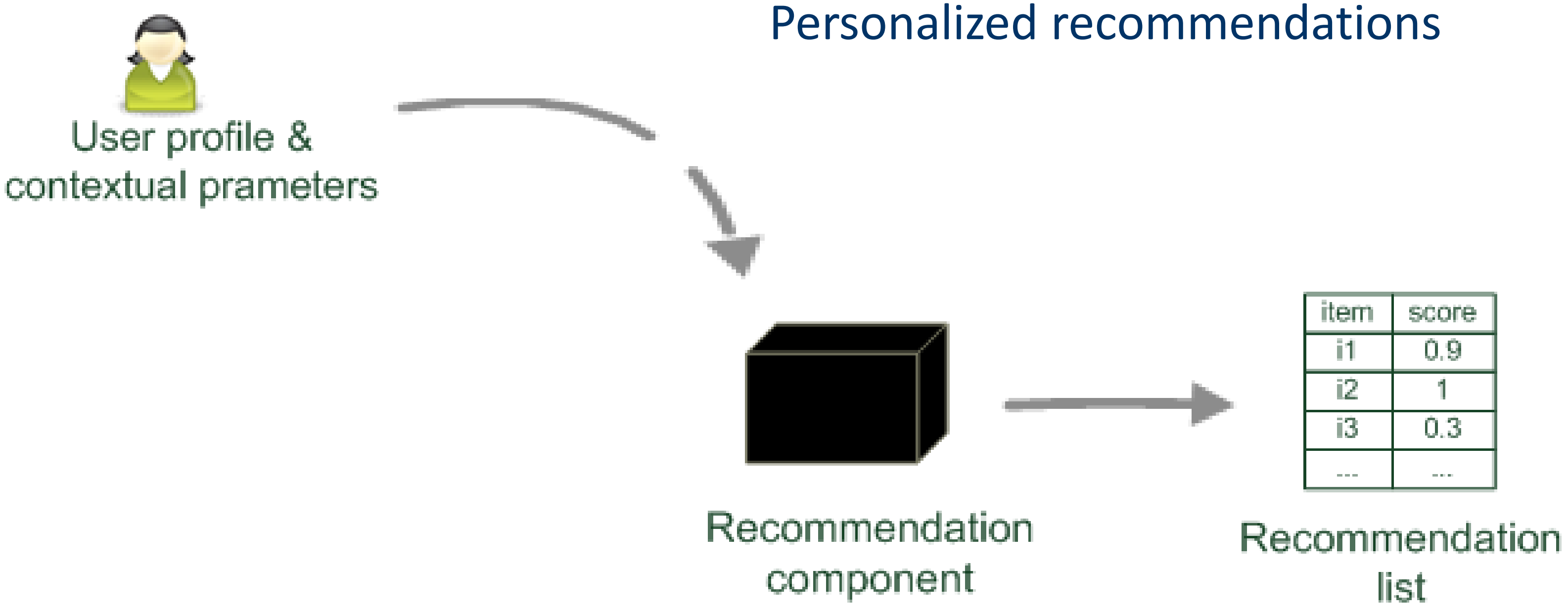
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance



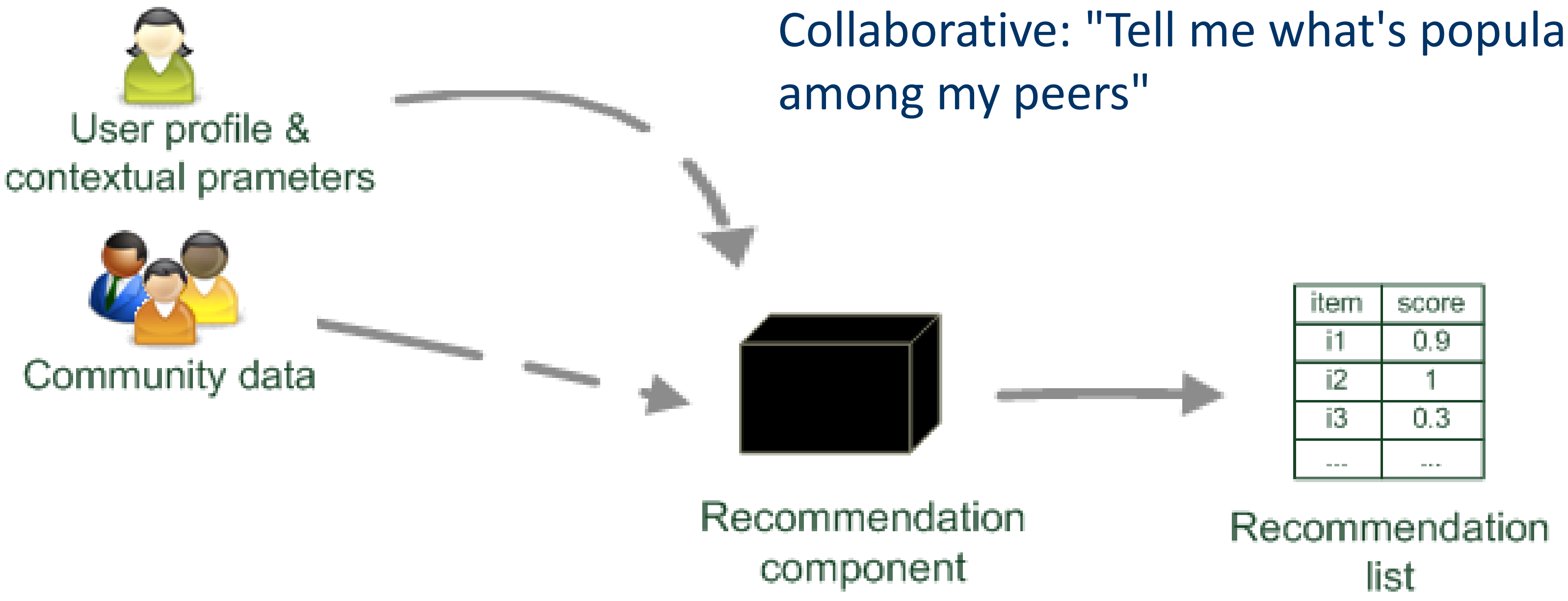
Paradigms of recommender systems

Personalized recommendations



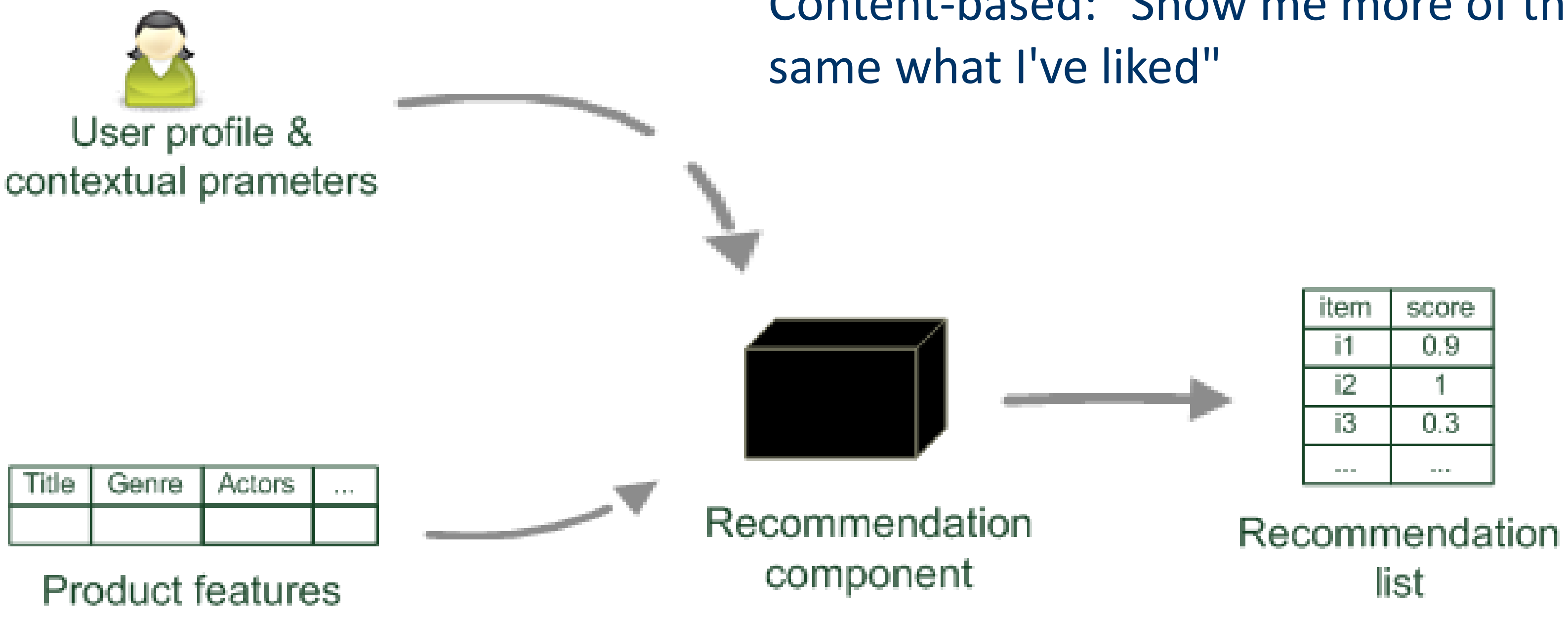
Paradigms of recommender systems

Collaborative: "Tell me what's popular among my peers"



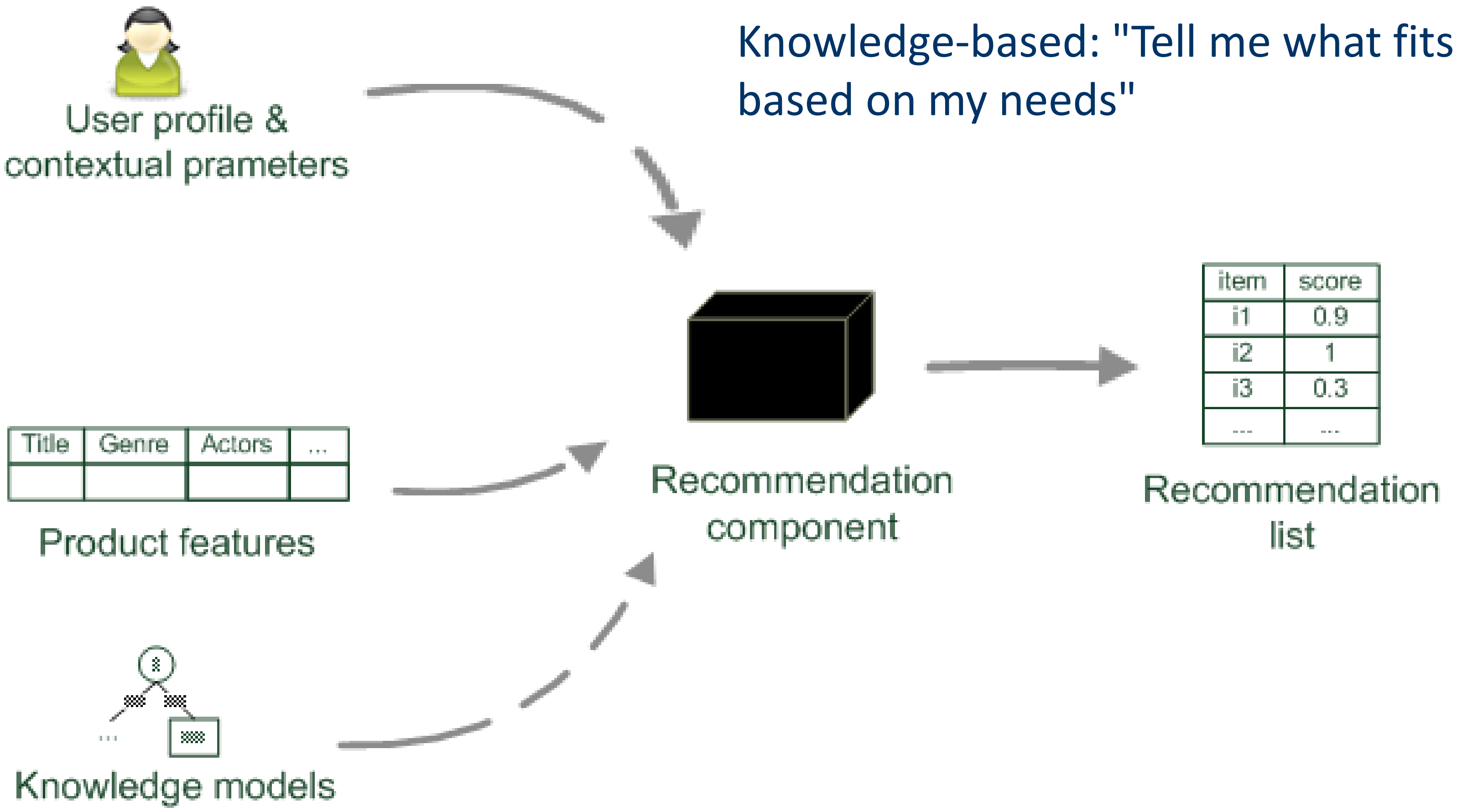
Paradigms of recommender systems

Content-based: "Show me more of the same what I've liked"



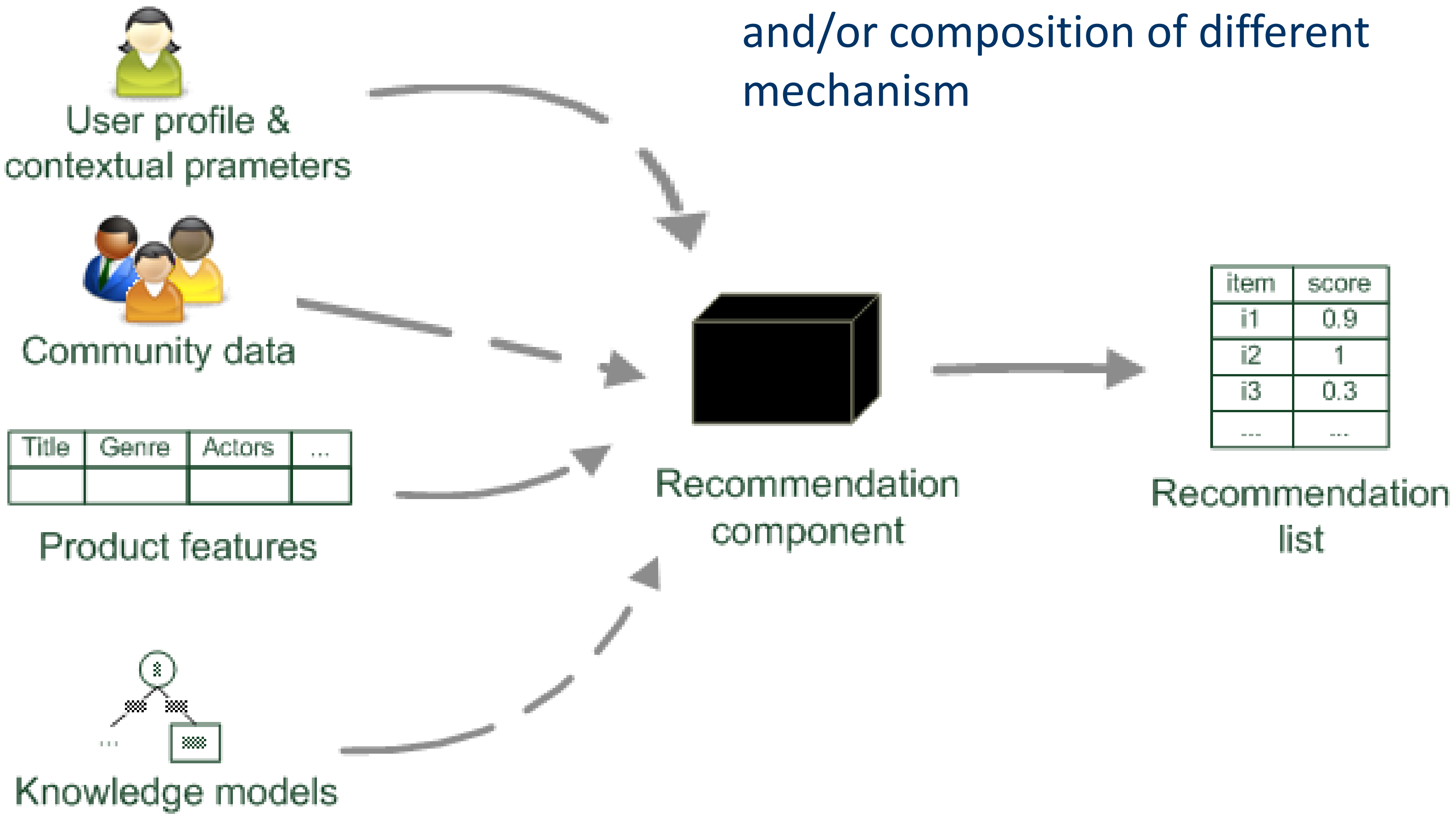
Paradigms of recommender systems

Knowledge-based: "Tell me what fits based on my needs"



Paradigms of recommender systems

Hybrid: combinations of various inputs and/or composition of different mechanism



Collaborative Filtering (CF)

- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)

- Approach
  - use the "wisdom of the crowd" to recommend items
- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

Pure CF Approaches

- Input
  - Only a matrix of given user–item ratings

- Output types
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
  - A top-N list of recommended items

User-based nearest-neighbor collaborative filtering (1)

- The basic technique
  - Given an "active user" (Alice) and an item i not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item i
    - use, e.g. the average of their ratings to predict, if Alice will like item i
    - do this for all items Alice has not seen and recommend the best-rated

- Basic assumption and idea
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

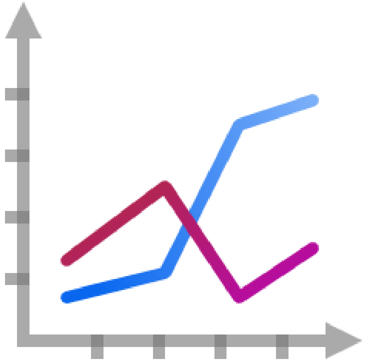
User-based nearest-neighbor collaborative filtering (2)

- Example
  - A database of ratings of the current user, Alice, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

User-based nearest-neighbor collaborative filtering (3)



- Some first questions
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Measuring user similarity

- A popular similarity measure in user-based CF: **Pearson correlation**

  a, b : users

  $r_{a,p}$: rating of user a for item p

  P : set of items, rated both by a and b

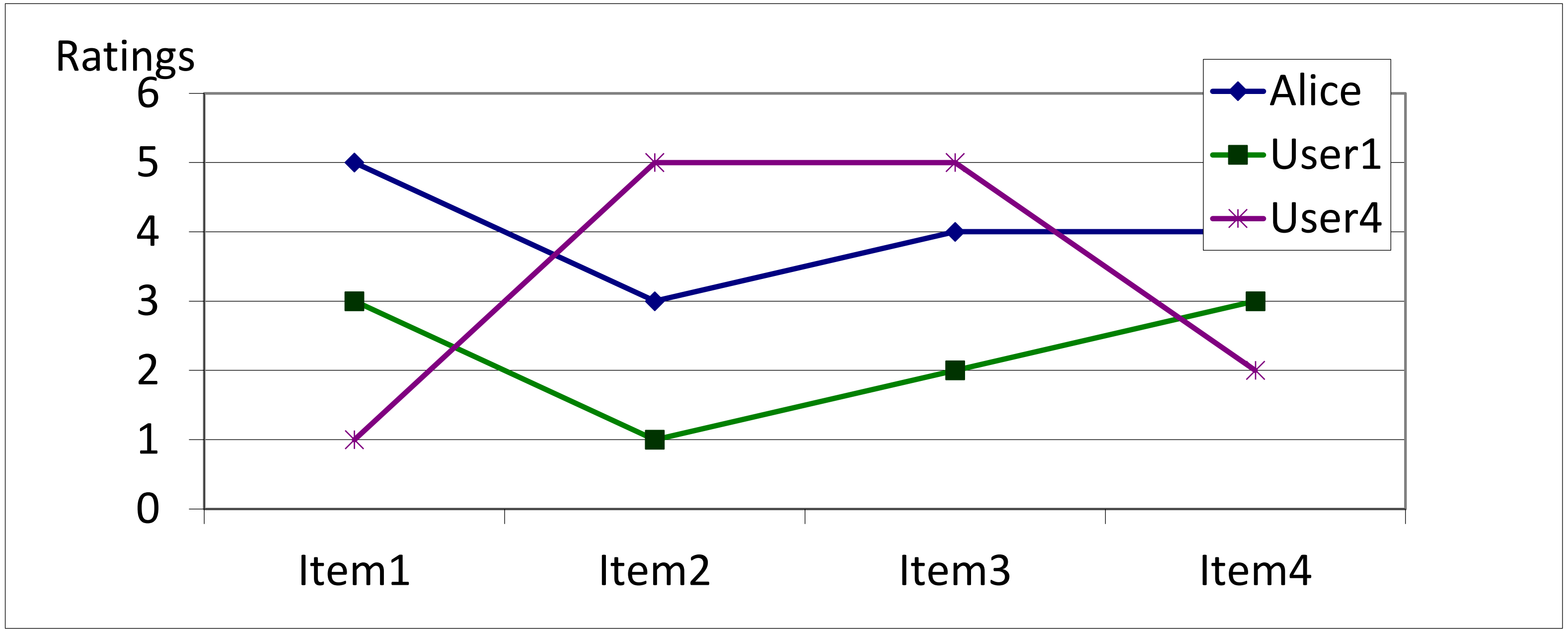
$$\boldsymbol{sim(a,b)} = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

  - Possible similarity values between -1 and 1

	Item1	Item2	Item3	Item4	Item5	
Alice	5	3	4	4	?	
User1	3	1	2	3	3	sim = 0.85
User2	4	3	4	3	5	sim = 0.70
User3	3	3	1	5	4	sim = 0.00
User4	1	5	5	2	1	sim = -0.79

Pearson correlation

- Takes differences in rating behavior into account



- Works well in usual domains, compared with alternative measures
  - such as cosine similarity

Making predictions

	Item1	Item2	Item3	Item4	Item5	
Alice	5	3	4	4	?	
User1	3	1	2	3	3	sim = 0.85
User2	4	3	4	3	5	sim = 0.70
User3	3	3	1	5	4	sim = 0.00
User4	1	5	5	2	1	sim = -0.79

- A common prediction function:

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b\in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b\in N} |sim(a,b)|}}$$

- Calculate, whether the neighbors' ratings for the unseen item i are higher or lower than their average
- Combine the rating differences – use the similarity with a as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Improving the metrics / prediction function

- Not all neighbor ratings might be equally "valuable"
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - **Possible solution**: Give more weight to items that have a higher variance

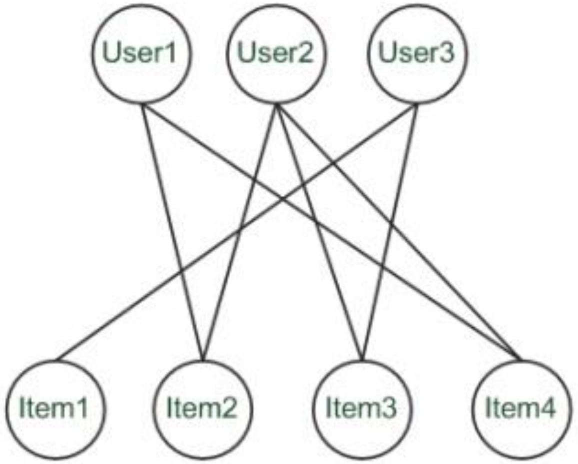
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors

Memory-based approaches

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items

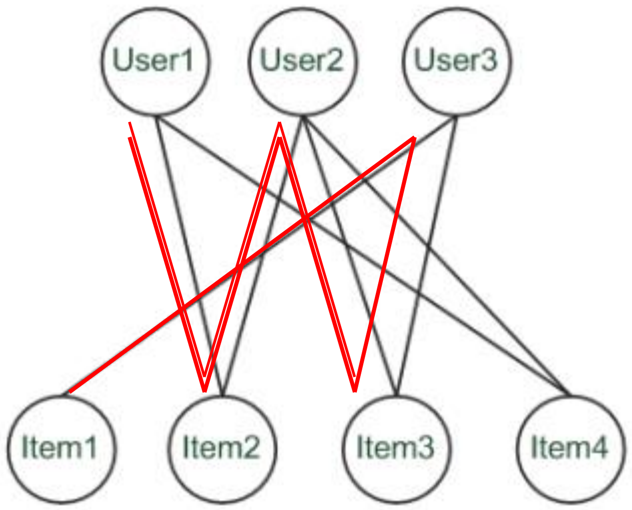
Graph-based methods (1)

- "Spreading activation" (Huang et al. 2004)
  - Exploit the supposed "transitivity" of customer tastes and thereby augment the matrix with additional information
  - Assume that we are looking for a recommendation for *User1*
  - When using a standard CF approach, *User2* will be considered a peer for *User1* because they both bought *Item2* and *Item4*
  - Thus *Item3* will be recommended to *User1* because the nearest neighbor, *User2*, also bought or liked it



Graph-based methods (2)

- "Spreading activation" (Huang et al. 2004)
  - In a standard user-based or item-based CF approach, paths of length 3 will be considered – that is, *Item3* is relevant for *User1* because there exists a three-step path (*User1–Item2–User2–Item3*) between them
  - Because the number of such paths of length 3 is small in sparse rating databases, the idea is to also consider longer paths (indirect associations) to compute recommendations
  - Using path length 5, for instance



Length 3: Recommend Item3 to User1
Length 5: Item1 also recommendable

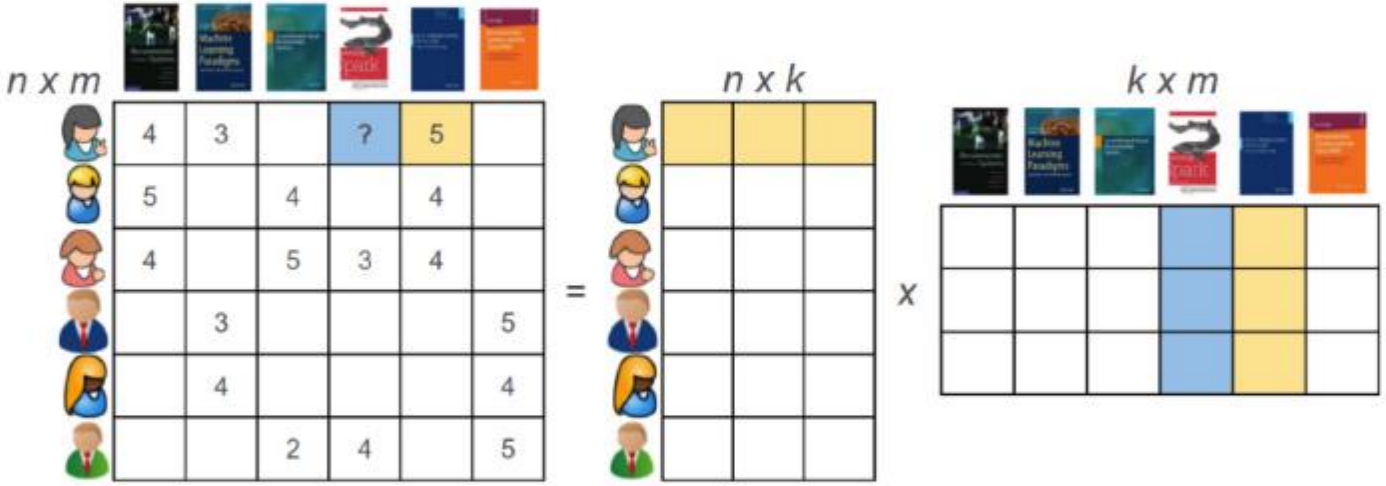
More model-based approaches

- Many techniques have been proposed e.g.,
  - Matrix factorization techniques, statistics
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches
- Costs of pre-processing
  - Usually not discussed
  - Incremental updates possible?

Matrix Factorization – A More General Formulation

- Matrix Factorization is one of the most popular methods for collaborative filtering
  - Given rating matrix Y
  - Each row represents an user *u*
  - While each column an item *i*

Regularization of p and q should be applied



$$\hat{y}_{ui} = f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}$$

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

2008: *Factorization meets the neighborhood: a multifaceted collaborative filtering model*, Y. Koren, ACM SIGKDD

- **Stimulated by work on Netflix competition**
  - Prize of $1,000,000 for accuracy improvement of 10% RMSE compared to own Cinematch system
  - Very large dataset (~100M ratings, ~480K users , ~18K movies)
  - Last ratings/user withheld (set K)
- **Root mean squared error metric optimized to 0.8567**
- **Metrics measure error rate**
  - Mean Absolute Error (*MAE*) computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i|$$

  - Root Mean Square Error (*RMSE*) is similar to *MAE*, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

Collaborative Filtering Issues

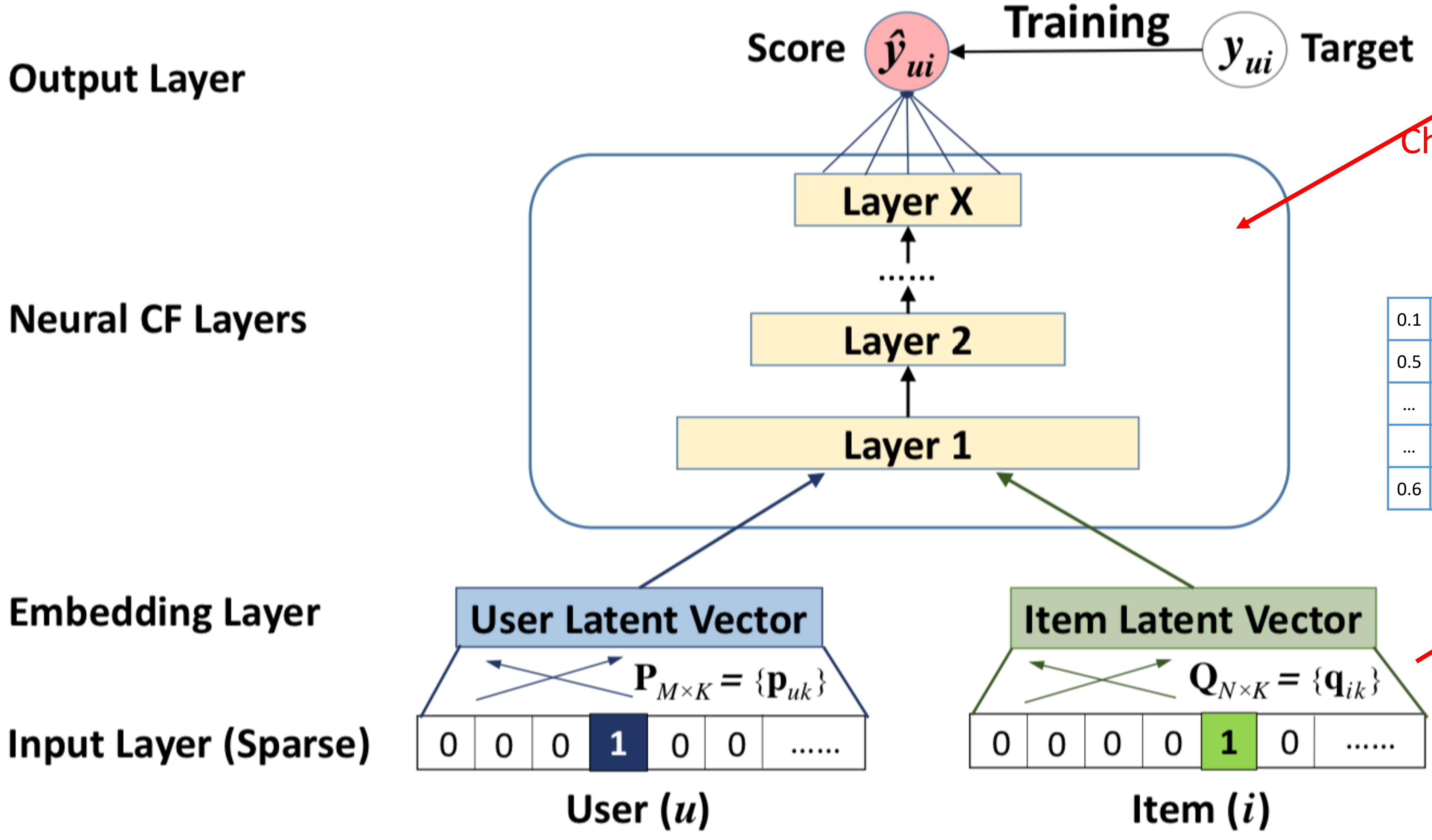
- Pros:
  - well-understood, works well in some domains, no knowledge engineering required
- Cons:
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- What is the best CF method?
  - In which situation and which domain? Inconsistent findings; always the same domains and data sets; differences between methods are often very small (1/100)
- How to evaluate the prediction quality?
  - MAE / RMSE: What does an MAE of 0.7 actually mean?
  - Diversity (novelty and surprising effect of recommendations)
    - Not yet fully understood
- What about multi-dimensional ratings?

Neural CF

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

$$\hat{y}_{ui} = \boxed{f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T\mathbf{q}_i = \sum_{k=1}^{K} p_{uk}q_{ik}}$$

Change to NN

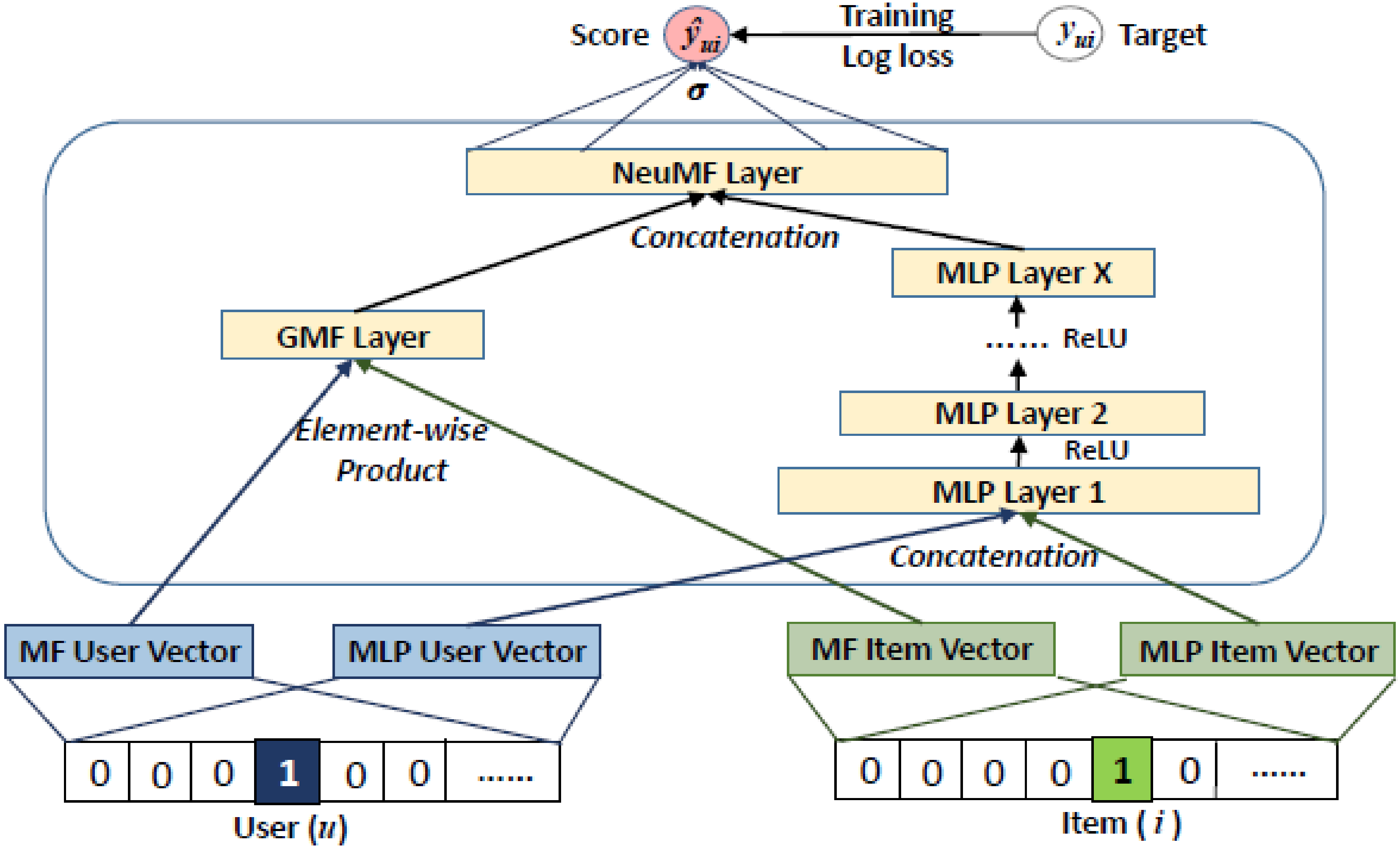


$$\begin{bmatrix} 0.1 & 2.4 & 1.6 & 1.8 & 0.5 & 0.9 & \dots & \dots & \dots & 3.2 \\ 0.5 & 2.6 & 1.4 & 2.9 & 1.5 & 3.6 & \dots & \dots & \dots & 6.1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0.6 & 1.8 & 2.7 & 1.9 & 2.4 & 2.0 & \dots & \dots & \dots & 1.2 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \\ \dots \\ \dots \\ 1.9 \end{bmatrix}$$

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua (2017). Neural Collaborative Filtering. In Proceedings of WWW '17, Perth, Australia, April 03-07, 2017.

More Advanced Model

Generalized Matrix Factorization (GMF)

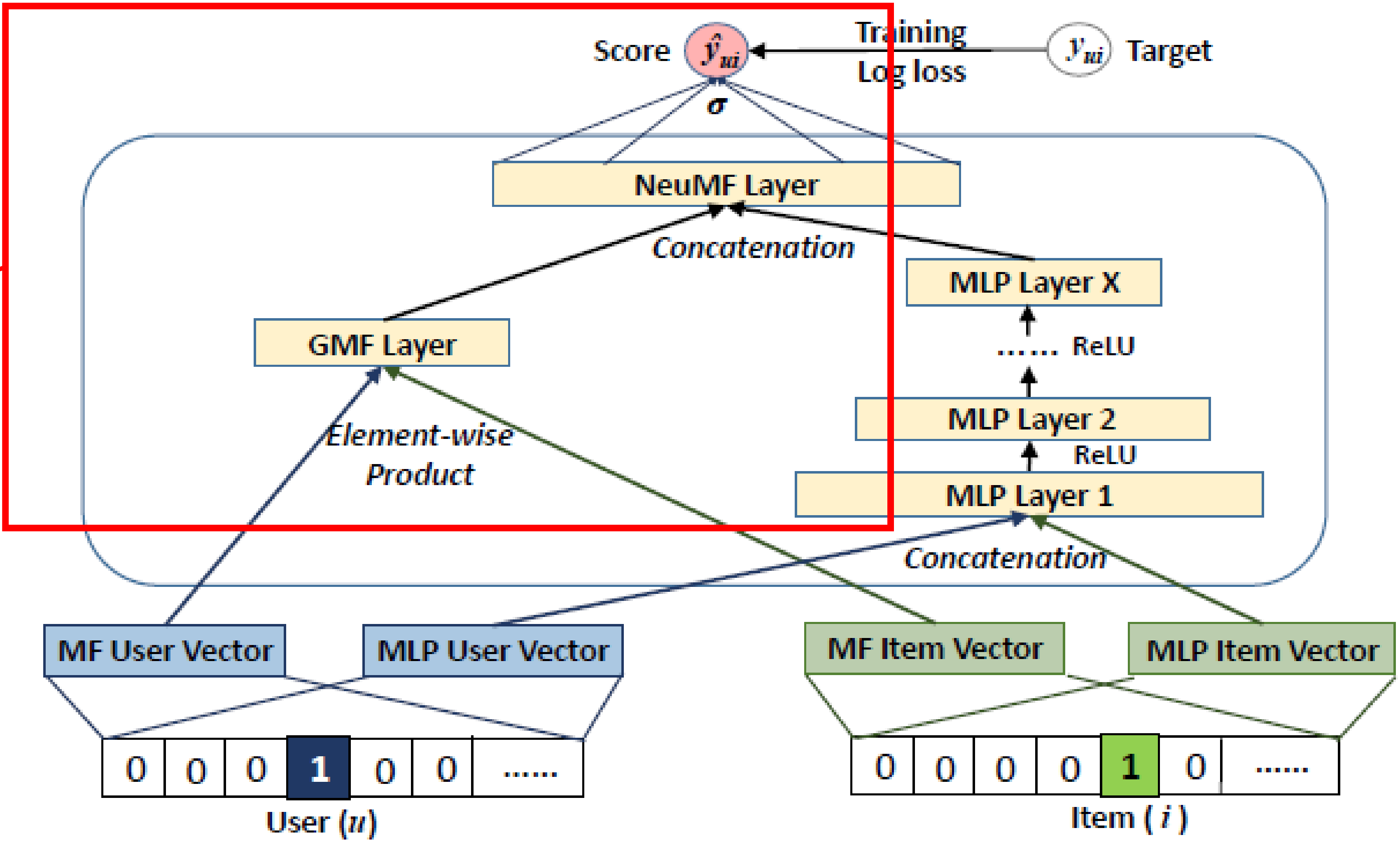


General Matrix Factorization (GMF)

$$\phi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i$$

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i))$$

sigmoid



Multi-layer Perceptron (MLP)

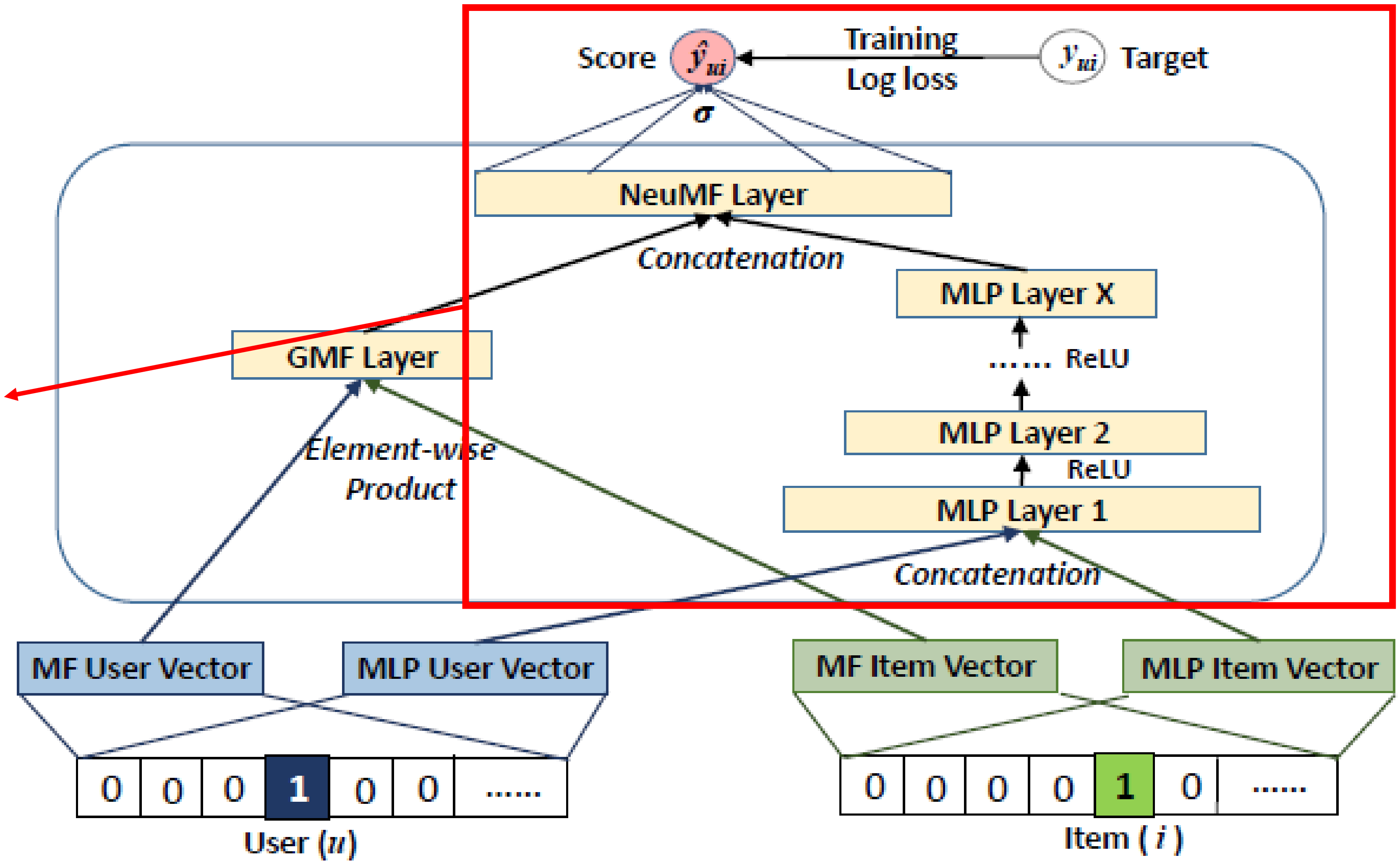
$$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix},$$

$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2),$$

$$\ldots\ldots$$

$$\phi_L(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),$$

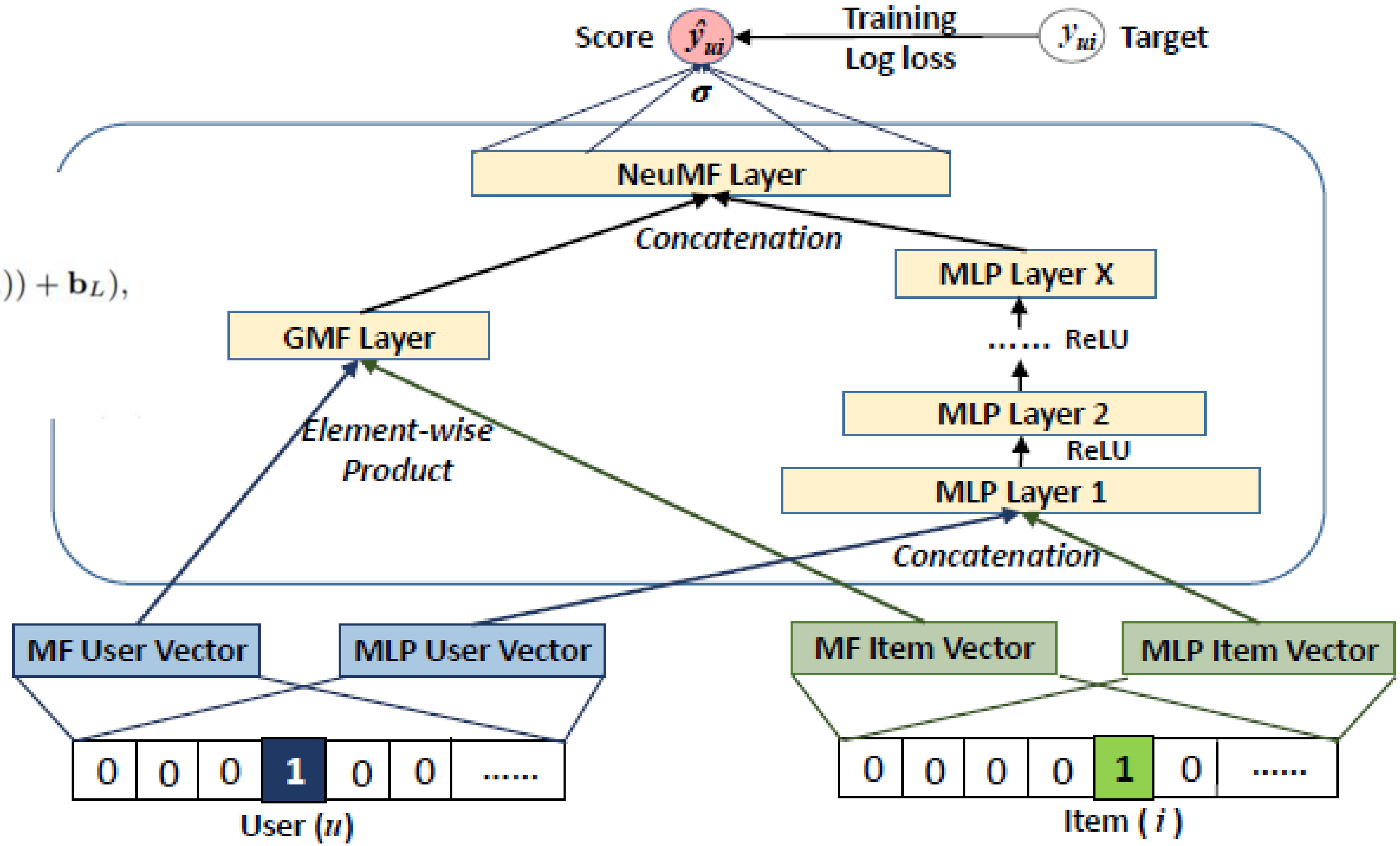


More Advanced Model

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G,$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix}\mathbf{p}_u^M \\ \mathbf{q}_i^M\end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix}\phi^{GMF} \\ \phi^{MLP}\end{bmatrix}),$$



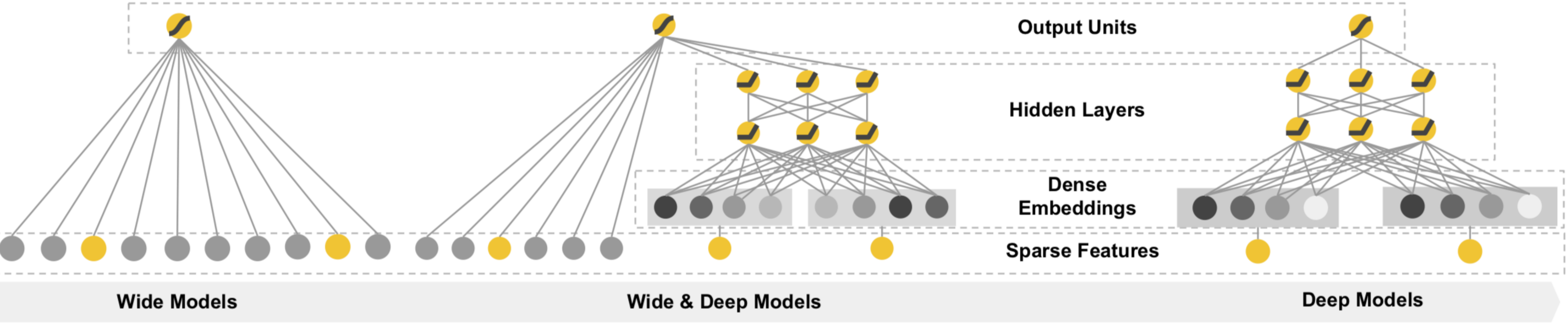
Allow GMF and MLP to learn separate embeddings

Deep learning for RS

- Neural Collaborative Filtering (NCF)

- Wide and Deep Learning (Google)
    - Combining dense features with categorical features

Wide & Deep Learning

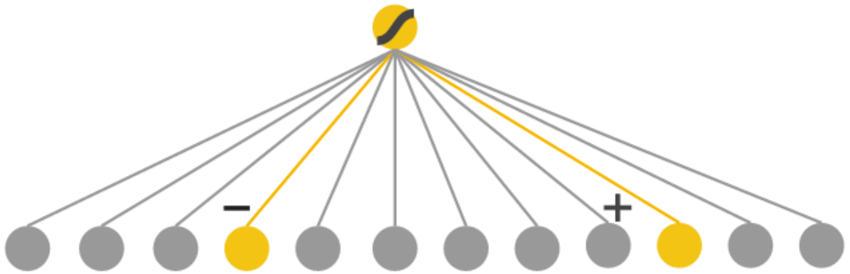


Memorization **Generalization**

https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html

Heng-Tze Cheng,Levent Koc,Jeremiah Harmsen,Tal Shaked,Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 7–10.

The wide model

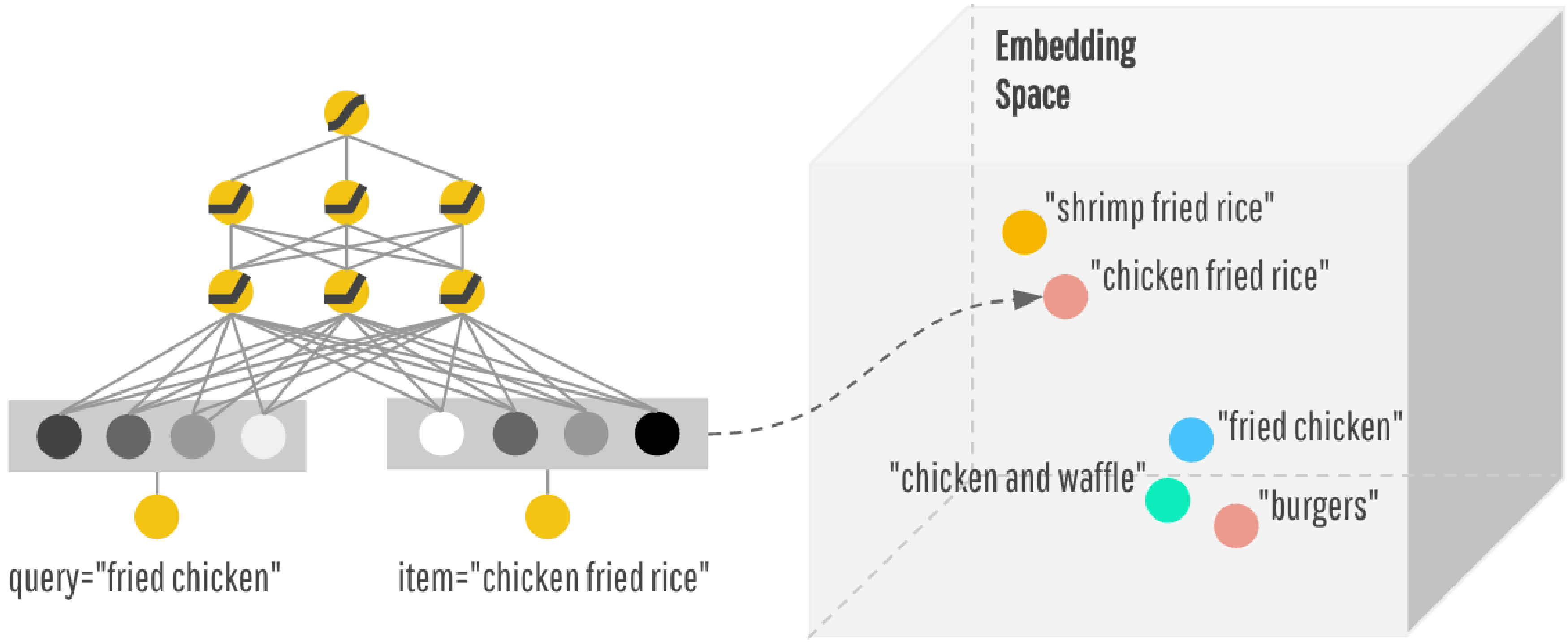


Linear model: $y = \mathbf{w}^T \mathbf{x} + b$ $\qquad$ $\mathbf{x} = [x_1, x_2, \dots, x_d]$ is a vector of d features

Cross-product transformation (traditional feature engineering) :

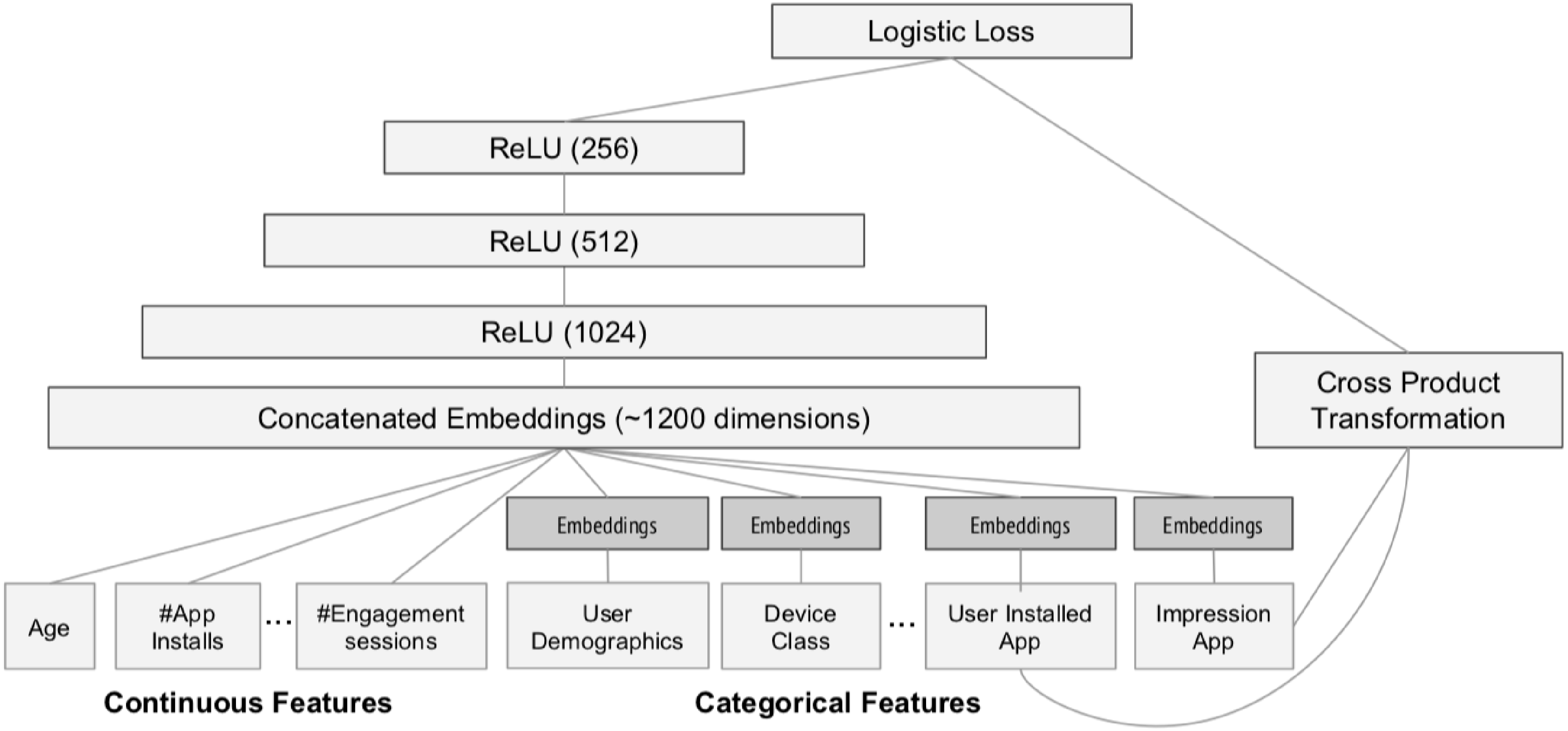
AND(user_installed_app=netflix, impression_app=pandora)

The deep model



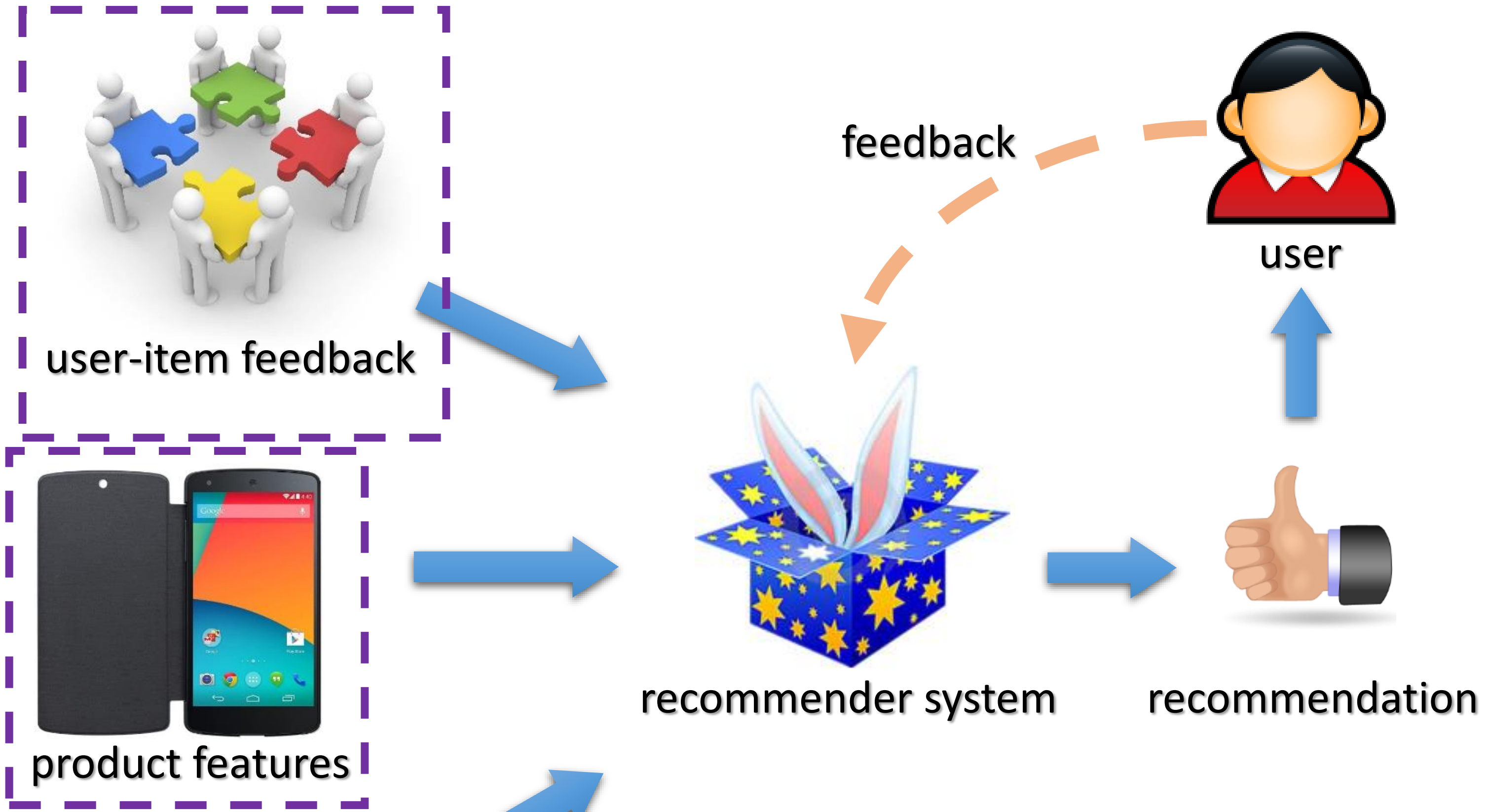
$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$$

The wide & deep model



$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^{T}[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^{T} a^{(l_f)} + b)$$

Recommendation Paradigm

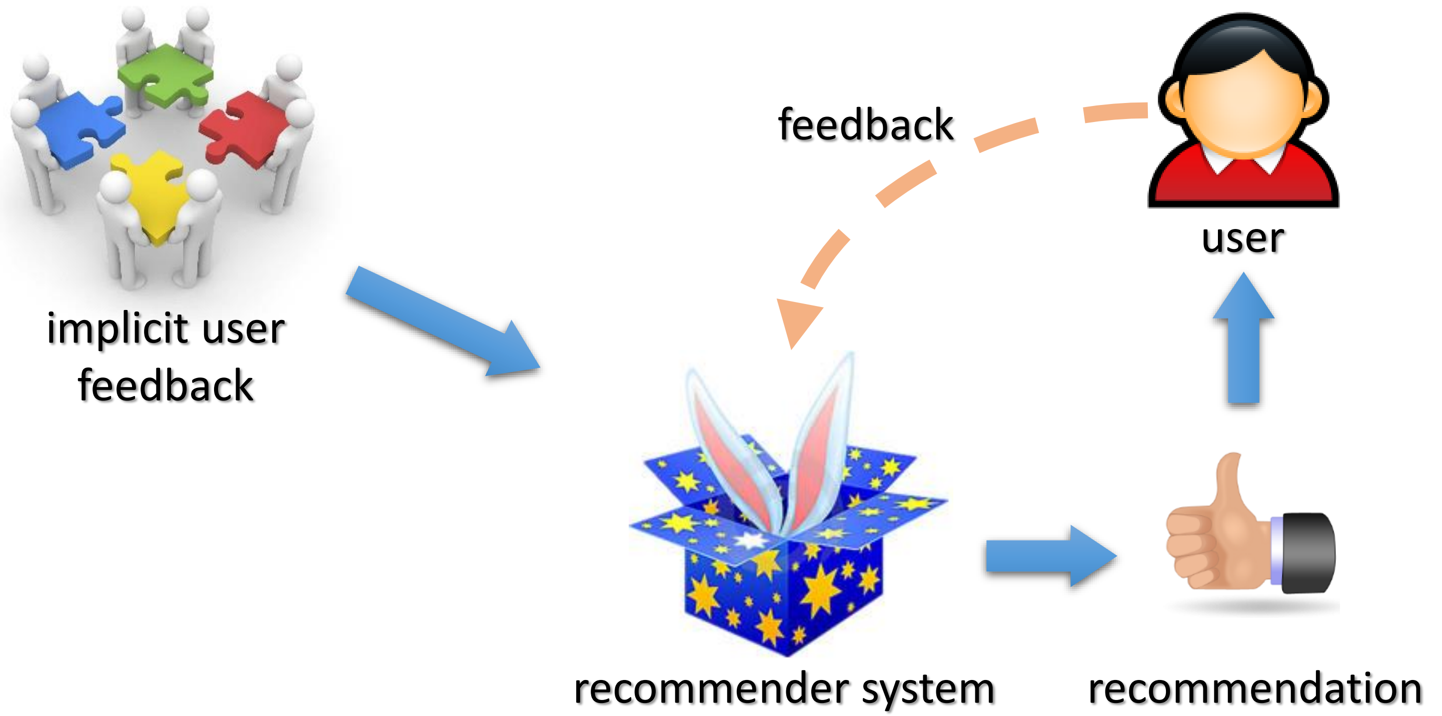


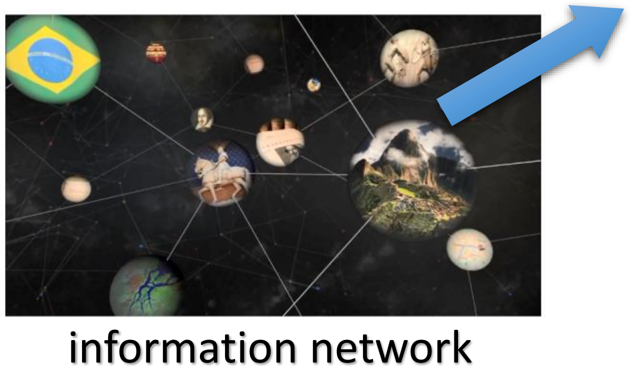
Collaborative Filtering
E.g., K-Nearest Neighbor (Sarwar WWW'01), Matrix Factorization (Hu ICDM'08, Koren IEEE-CS'09), Probabilistic Model (Hofmann SIGIR'03)

Content-Based Filtering
(Balabanovic Commun. ACM'97, Zhang SIGIR'02)

Hybrid Methods
E.g., Content-Based CF, External Knowledge CF (Ma WSDM'11)

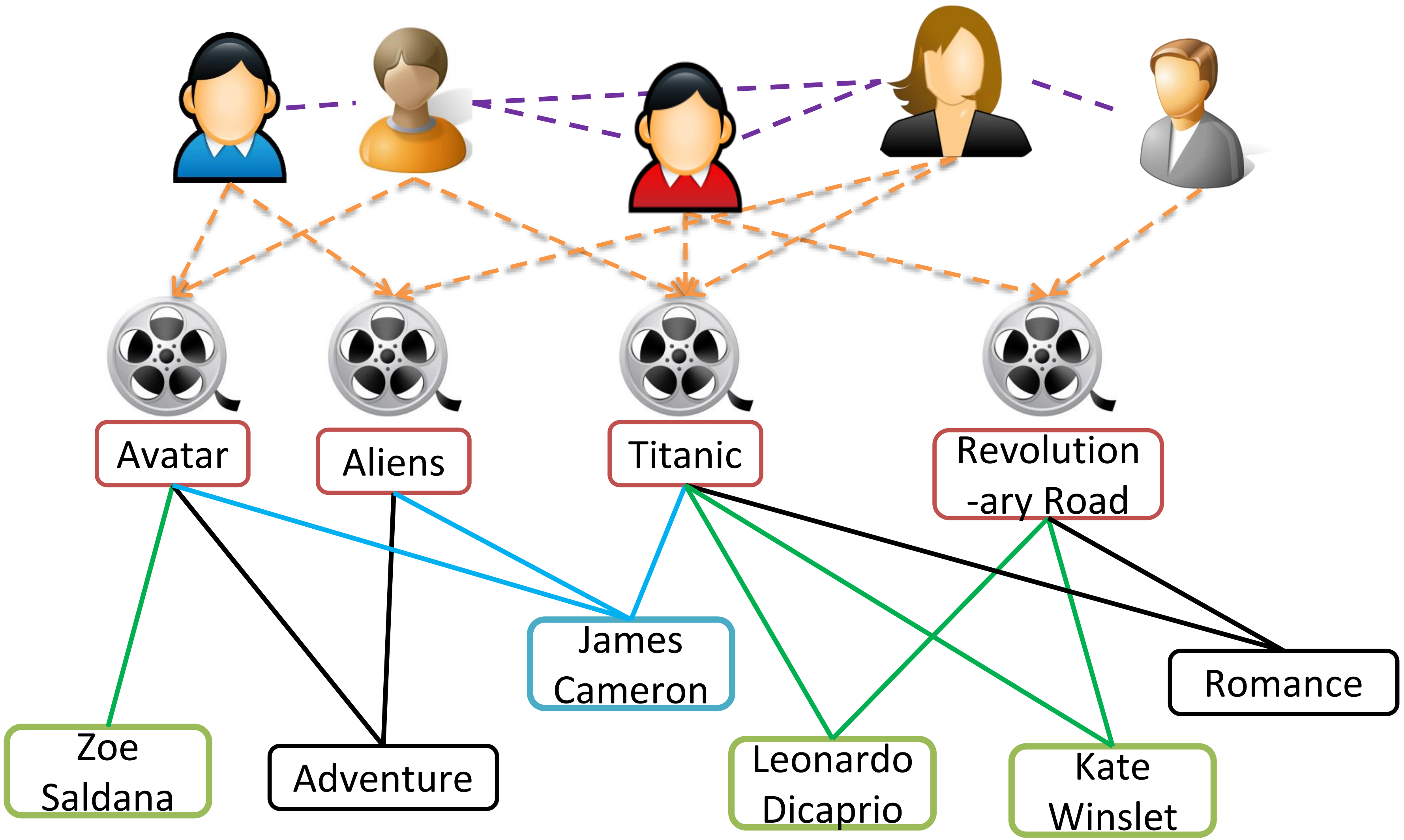
Problem Definition





hybrid collaborative filtering with information networks

The Heterogeneous Information Network View of Recommender System

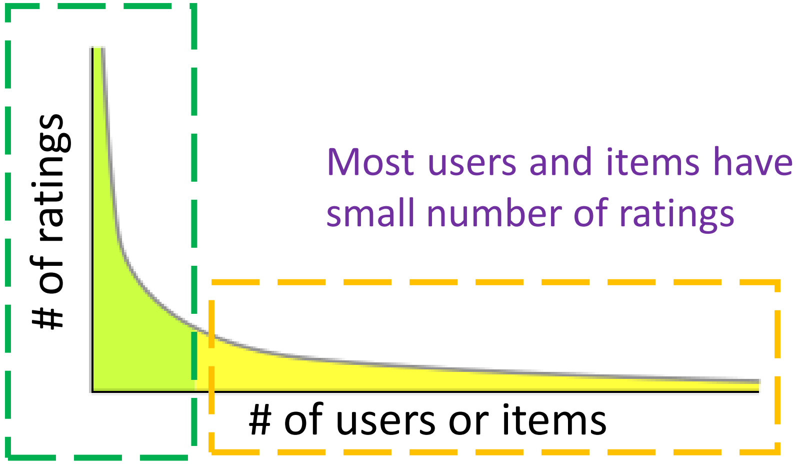


Relationship Heterogeneity Alleviates Data Sparsity

Collaborative filtering methods suffer from data sparsity issue

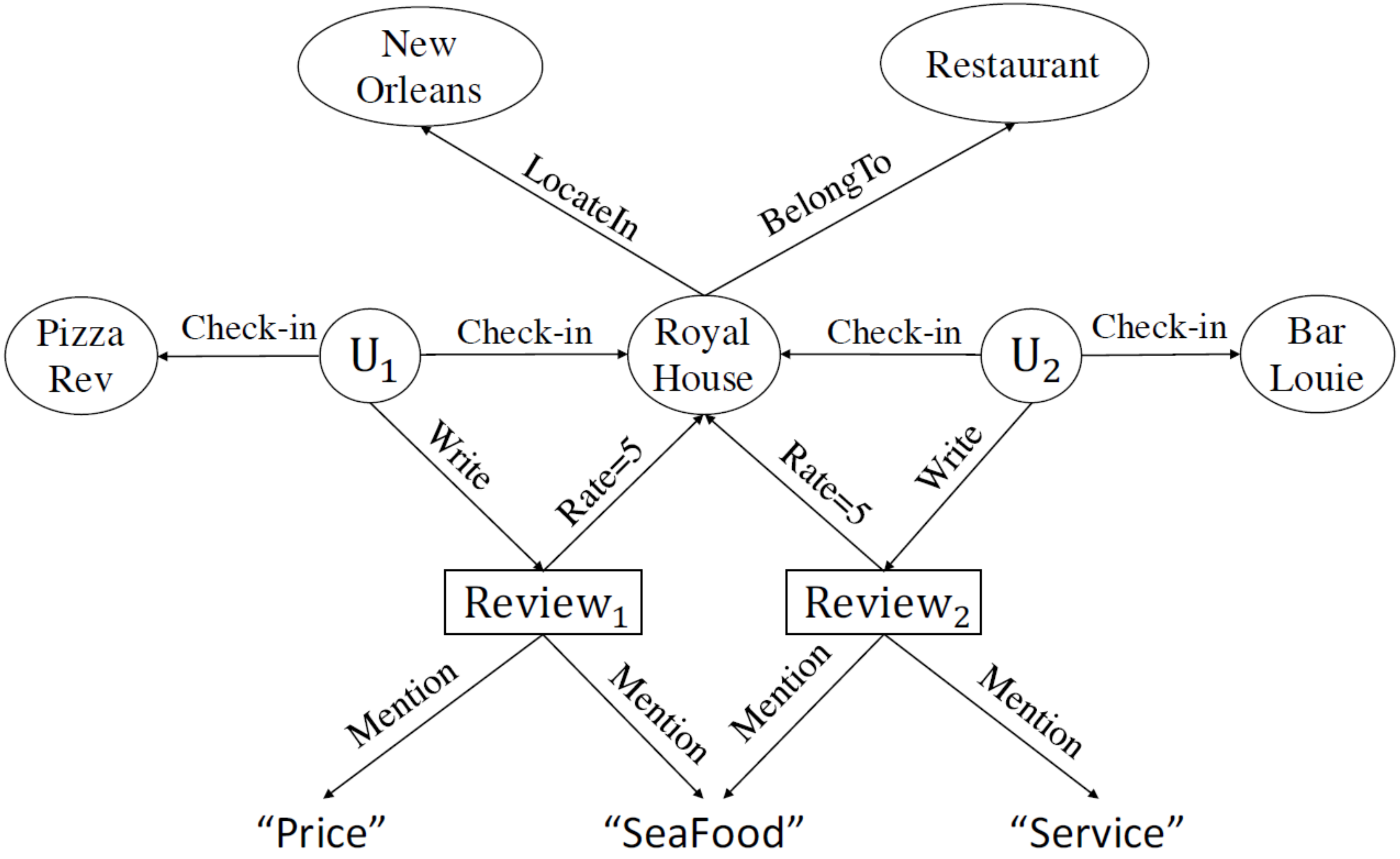
A small number of users and items have a large number of ratings

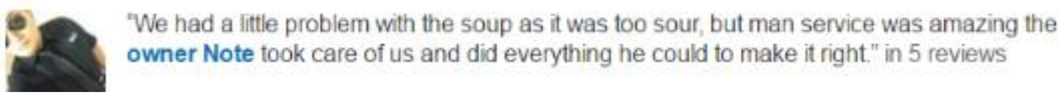
Most users and items have a small number of ratings

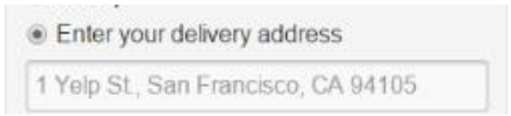


- Heterogeneous relationships complement each other
- Users and items with limited feedback can be connected to the network by different types of paths
  - Connect new users or items (cold start) in the information network

Yelp: A Heterogeneous Information Network

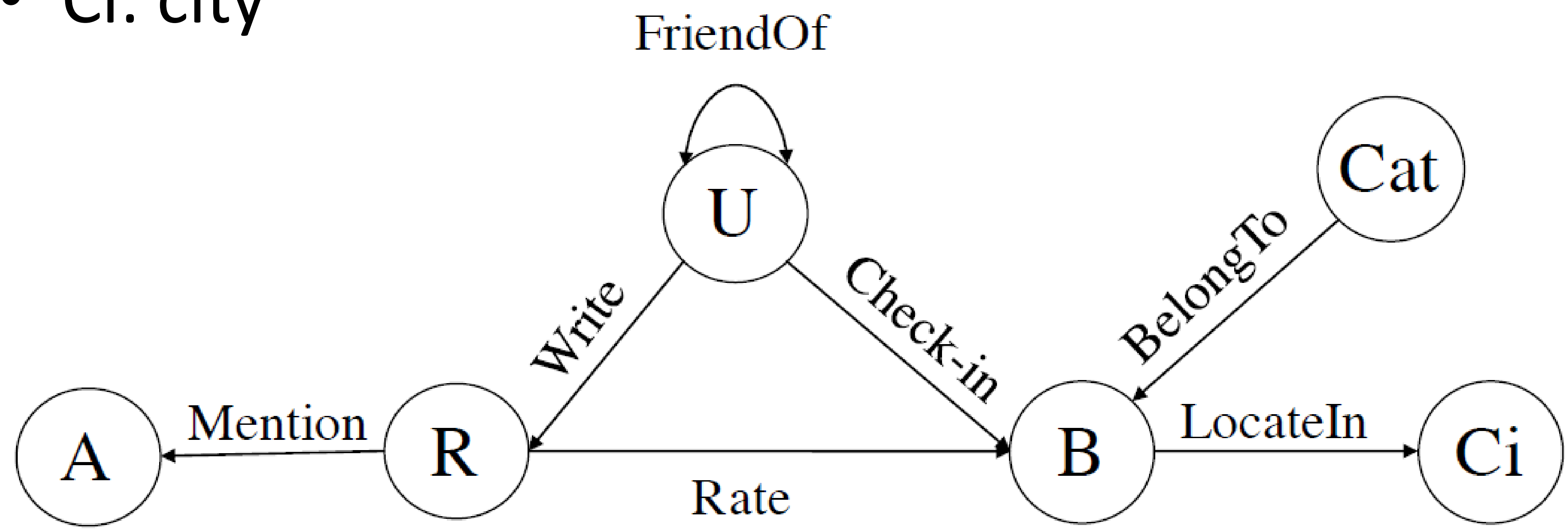




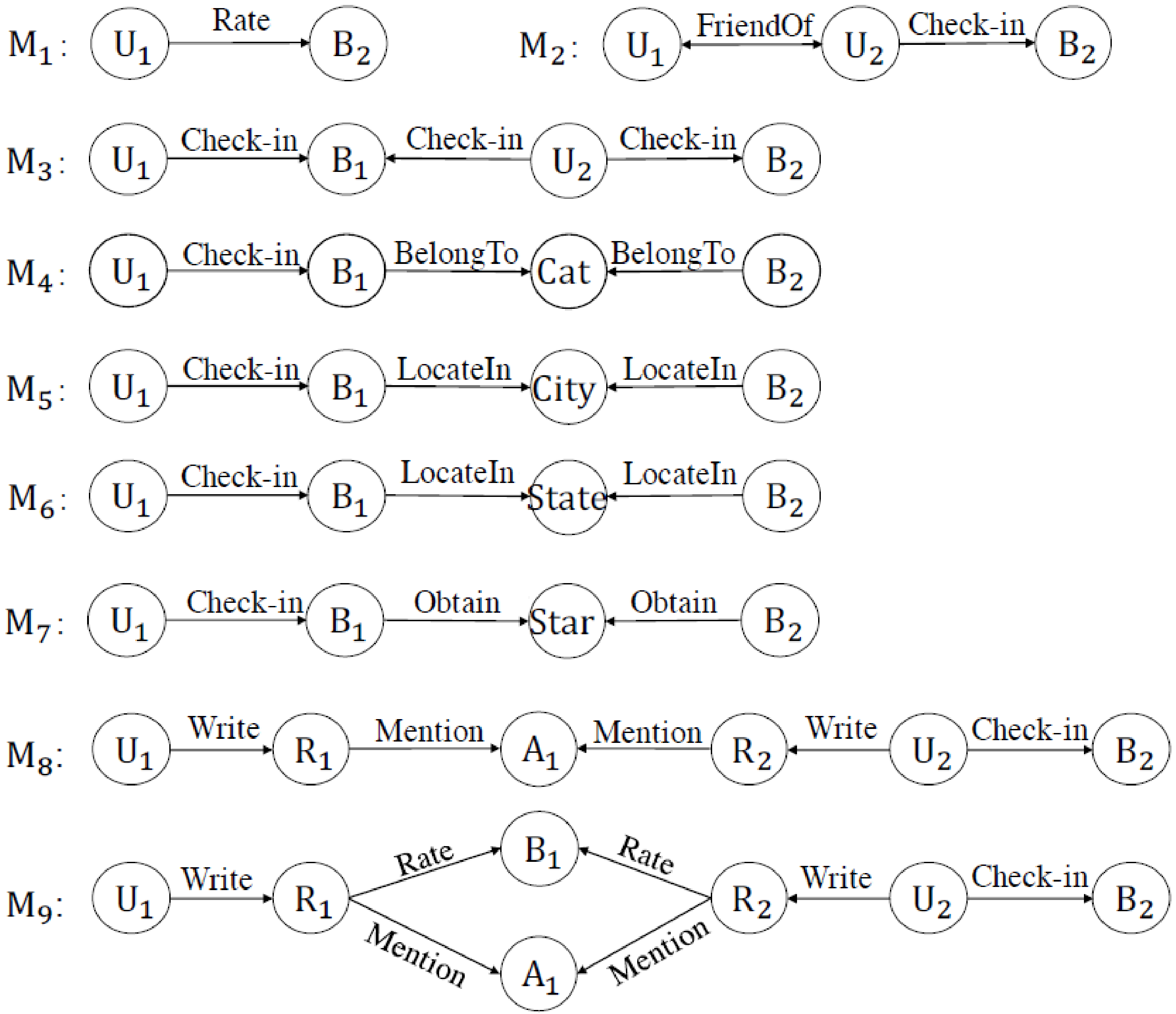


A Typical Network Schema of Yelp

- R: reviews;
- U: users;
- B: business;
- Cat: category of item;
- Ci: city



Meta-paths/graphs Extracted From Yelp



Matrix Formulation: Traditional CF

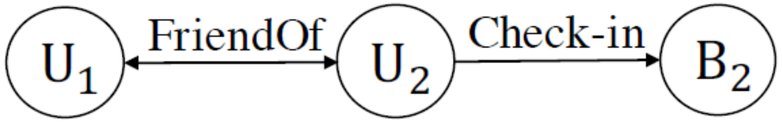
$$M_1: U_1 \xrightarrow{\text{Rate}} B_2$$

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	2		3		4		
u_2	4	3			5			
u_3	4		2				2	4
u_4								
u_5	5	1	2		4	3		
u_6	4	3		2	4		3	5

$\mathbf{W}_{UB}$

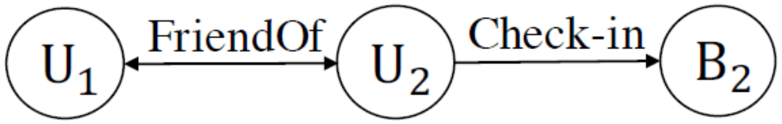
PathCount: Meta-path based similarities

- Number of meta-path instances connecting users and items



- Matrix multiplication.
  - W_{UB}: number of instances between type U and type B
  - W_{UU}: number of instances between type U and type U
    - Whether two users are friends
  - $W_{UU}\, W_{UB}$

Example



	u_1	u_2	u_3
u_1		1	
u_2	1		1
u_3		1	

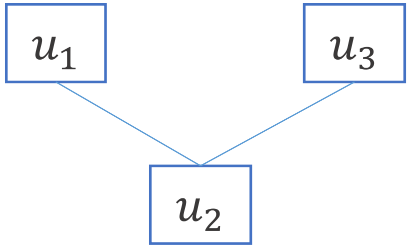
W_{UU}

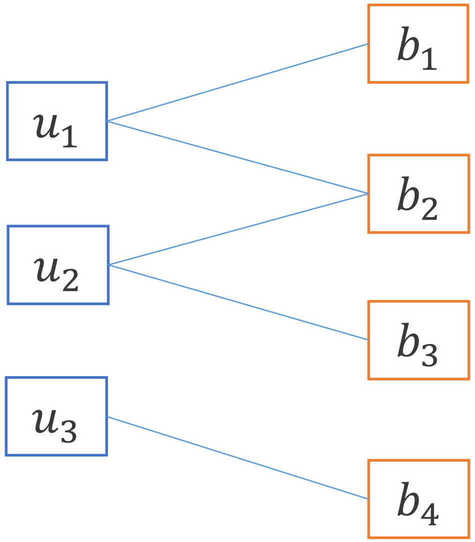
	b_1	b_2	b_3	b_4
u_1	1	1		
u_2		1	1	
u_3				1

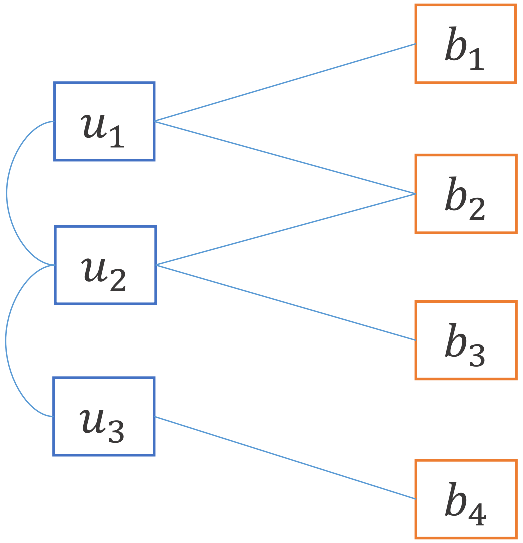
W_{UB}

	b_1	b_2	b_3	b_4
u_1		1	1	
u_2	1	1		1
u_3		1	1	

$W_{UU}\ W_{UB}$

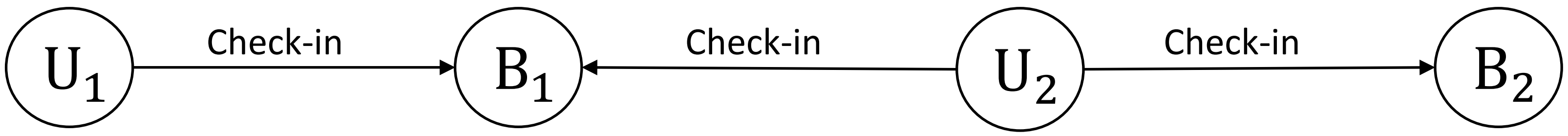




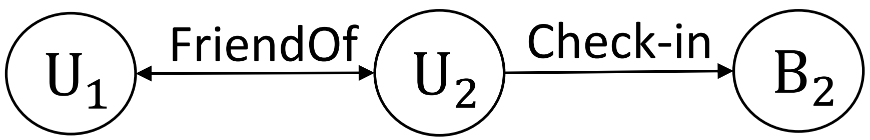


Metapath based RS

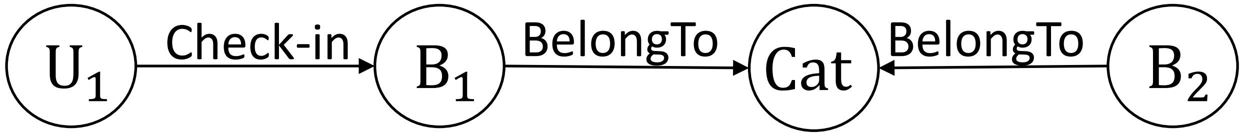
- Metapath → Recommending Strategy.



User-based CF

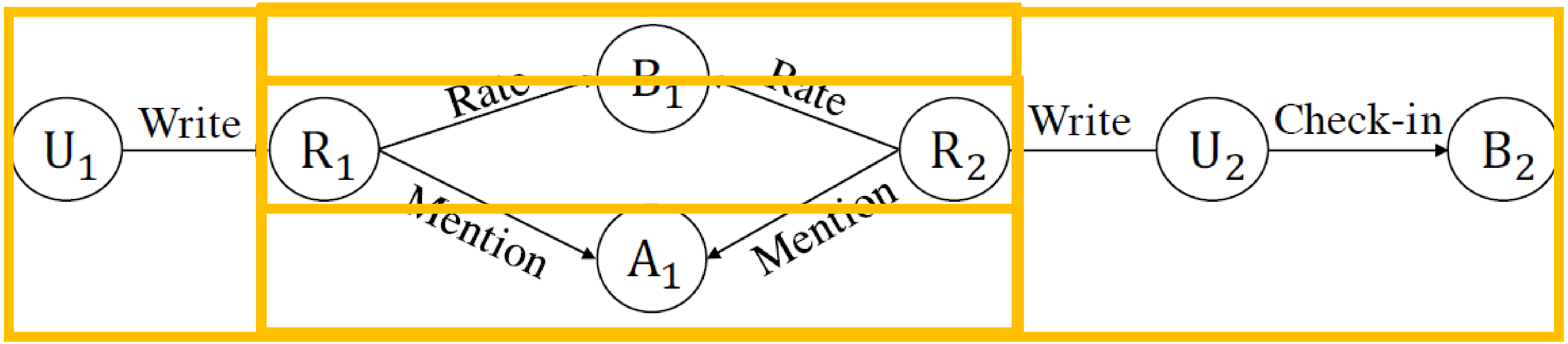


Social Recommendation



Content-based recommendation

Compute Similarity based on a Meta-graph



Compute $\mathbf{C}_{P_1}$: $\mathbf{C}_{P_1} = \mathbf{W}_{RB} \cdot \mathbf{W}_{RB}^\top;$

Compute $\mathbf{C}_{P_2}$: $\mathbf{C}_{P_2} = \mathbf{W}_{RA} \cdot \mathbf{W}_{RA}^\top;$

Compute $\mathbf{C}_{S_r}$: $\mathbf{C}_{S_r} = \mathbf{C}_{P_1} \odot \mathbf{C}_{P_2};$

Compute $\mathbf{C}_{M_9}$: $\mathbf{C}_{M_9} = \mathbf{W}_{UR} \cdot \mathbf{C}_{S_r} \cdot \mathbf{W}_{UR}^\top \cdot \mathbf{W}_{UB};$