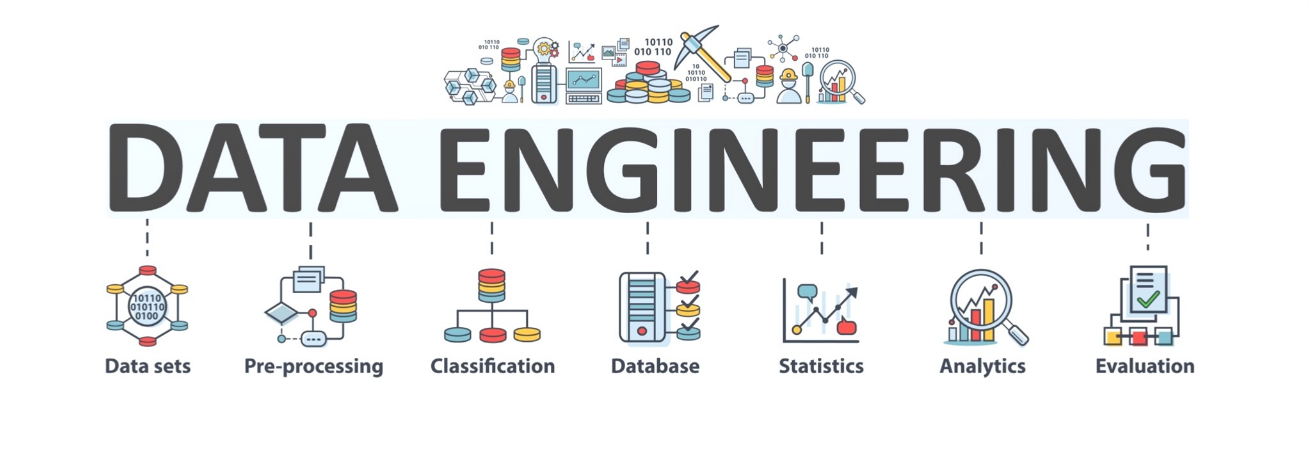


Lecture 7: Data Driven Applications (1) --

Information Retrieval and Recommendations

CS5481 Data Engineering

Instructor: Linqi Song

Outline

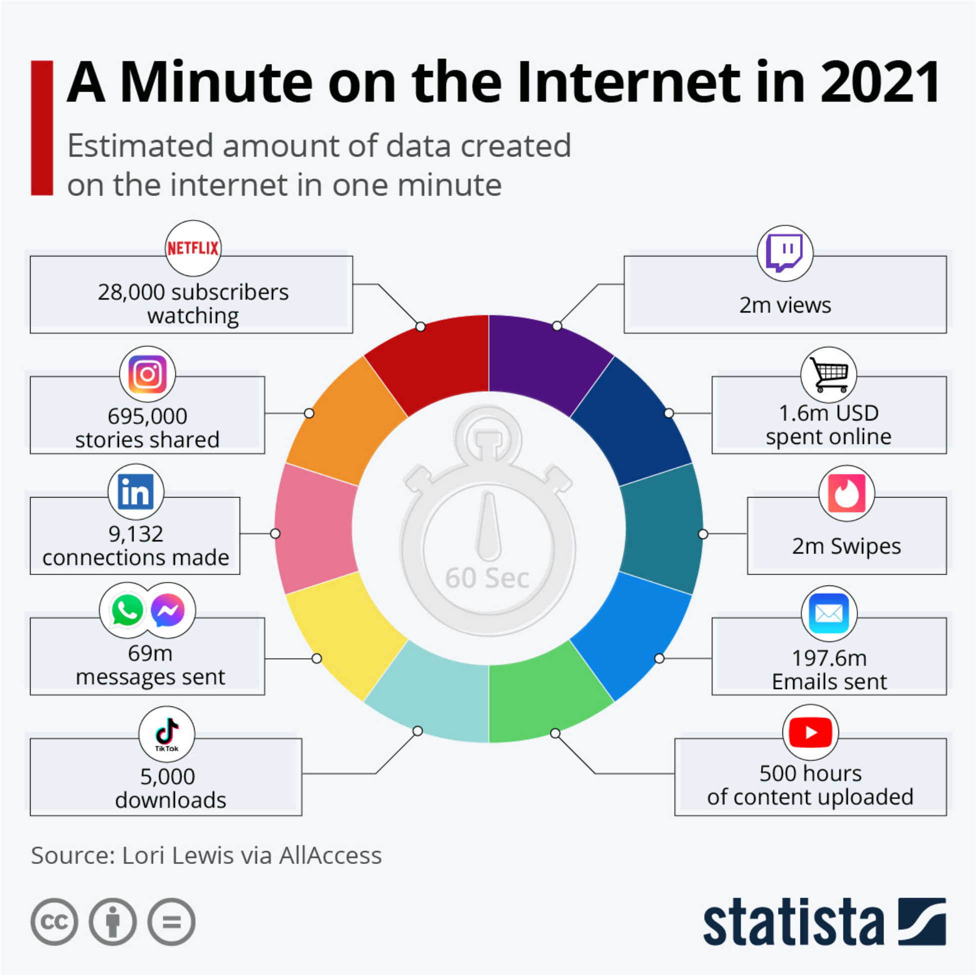
1. Information filtering

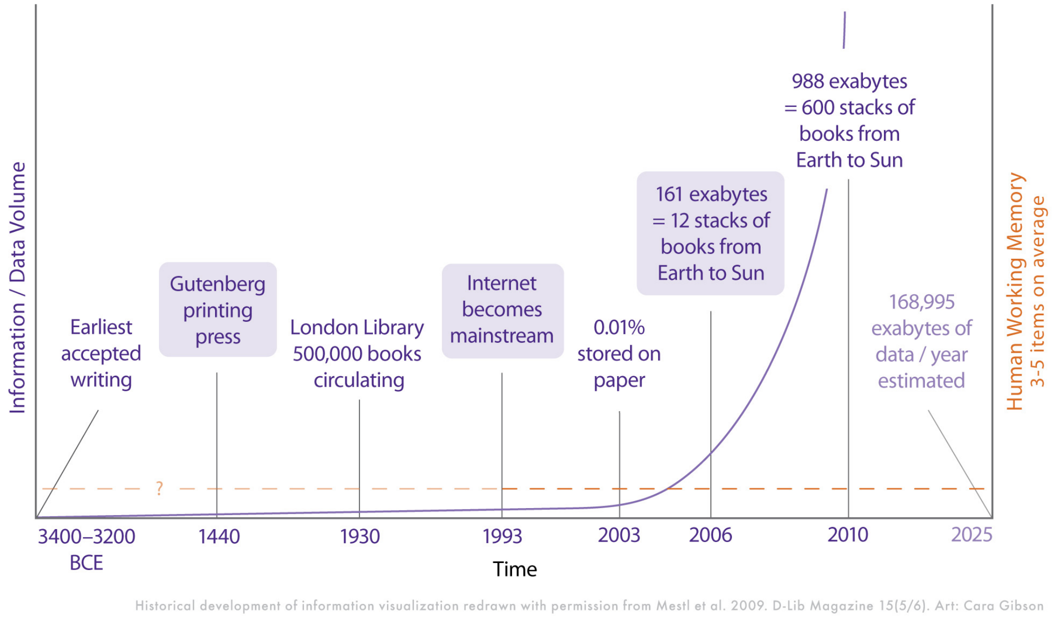
2. Information retrieval – process, approaches

3. Recommendations – process, approaches

Information overload

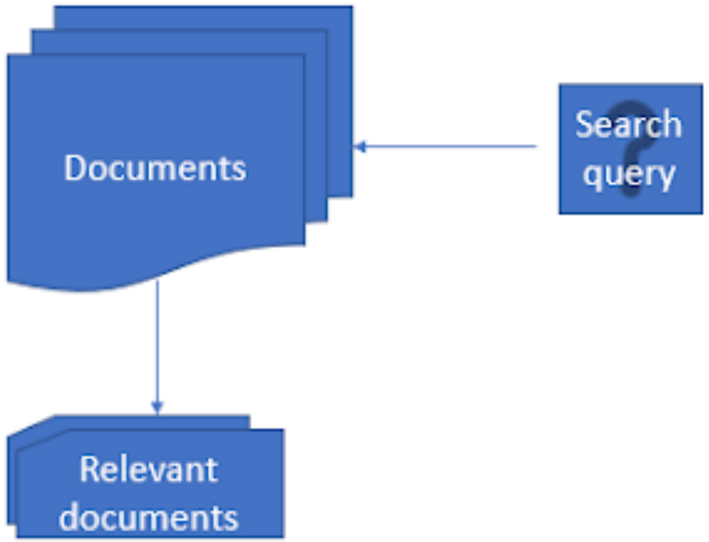
- Information overload is a state of being overwhelmed by the amount of data presented for one's attention or processing.



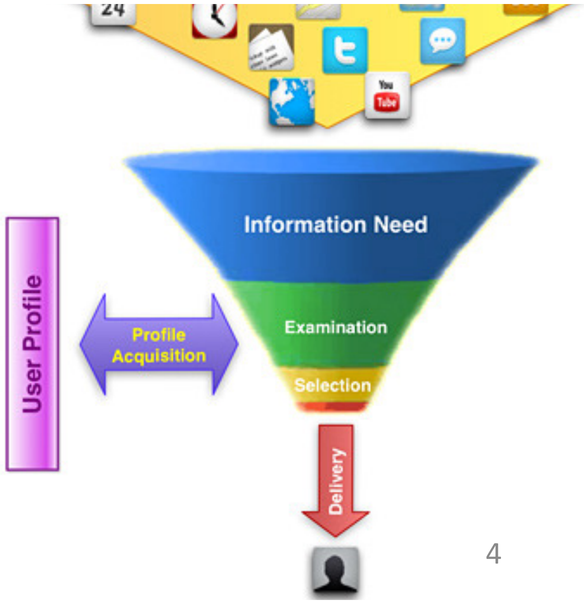


To overcome information overload -- information retrieval and information filtering

- Information retrieval (e.g., search engines): obtaining information system resources that are relevant to an information need (usually for a query) from a collection of those resources.



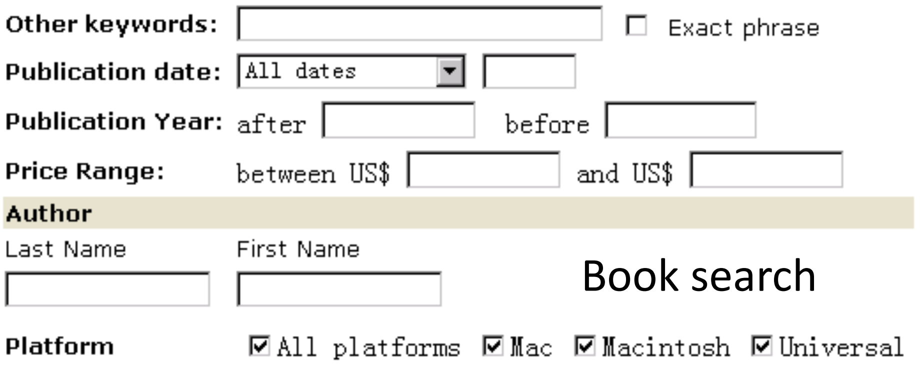
- Information filtering (e.g., recommender systems): the process of filtering information relevant to an individual's information needs, in order to come to terms with the ever increasing amount of information.



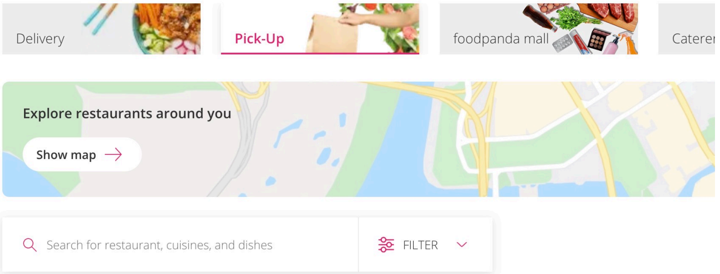
Information retrieval (IR)

- It is ‘search’: mostly searching for documents, but can also be others.
- It is a computer science discipline that designs and implements algorithms and tools to help people find information that they want.
  - ❑ From one or multiple large collections of materials (text or multimedia, structured or unstructured, with or without hyperlinks, with or without metadata, in a foreign language or not),
  - ❑ Where people can be a single user or a group.
  - ❑ Who initiate the search process by an information need, and, the resulting information should be relevant to the information need (based on the judgement by the person who starts the search).

Web search



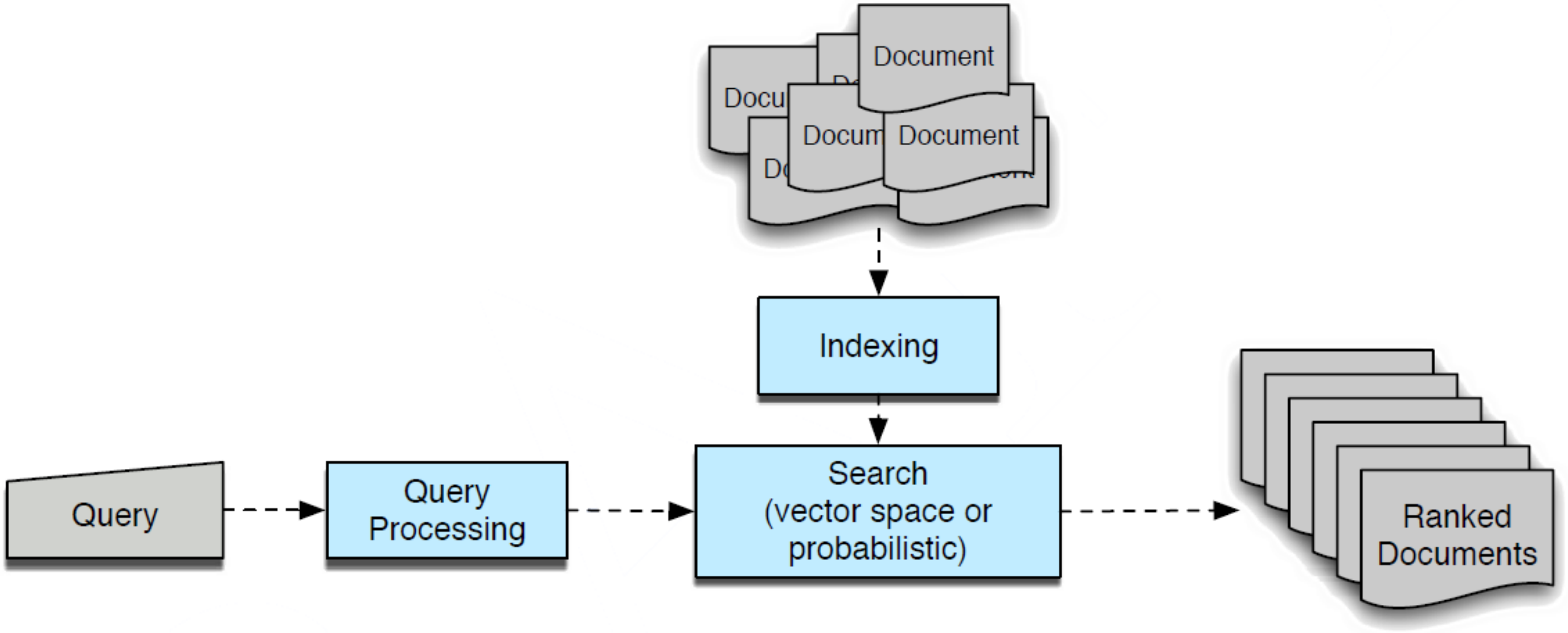
Book search



Food search

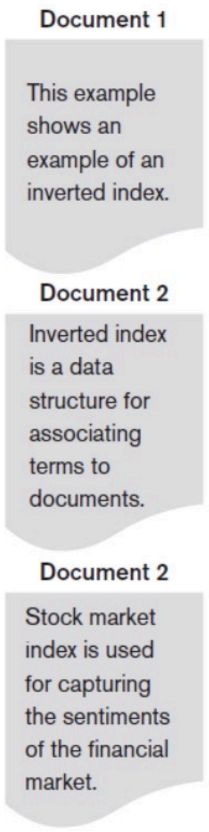
Process of information retrieval

- Some of the key issues
  - Information need: query and query processing
  - Relevance: indexing documents and search algorithm
  - Evaluation: document ranking, precision-recall, etc.



Indexing

- Document indexing: tags documents with certain attributes or labels that can later be efficiently searched through and retrieved.
- Build the vocabulary: set of distinct query terms in the document set.
- Use inverted index: data structure that attaches distinct terms with a list of all documents that contains the term



ID	Term	Document: position
1.	example	1:2, 1:5
2.	inverted	1:8, 2:1
3.	index	1:9, 2:2, 3:3
4.	market	3:2, 3:13

How to find relevant documents for a query?

- By keyword matching
  - ❑Boolean model
- By similarity
  - ❑Vector space model
- By imaging how to write out a query
  - ❑How likely a query is written with this document in mind.
  - ❑Generate with some randomness
  - ❑Query generation language model
- By trusting how other web pages think about the web page
  - ❑Pagerank, hits
- By trusting how other people find relevant documents for the same or similar query
  - ❑Learning to rank

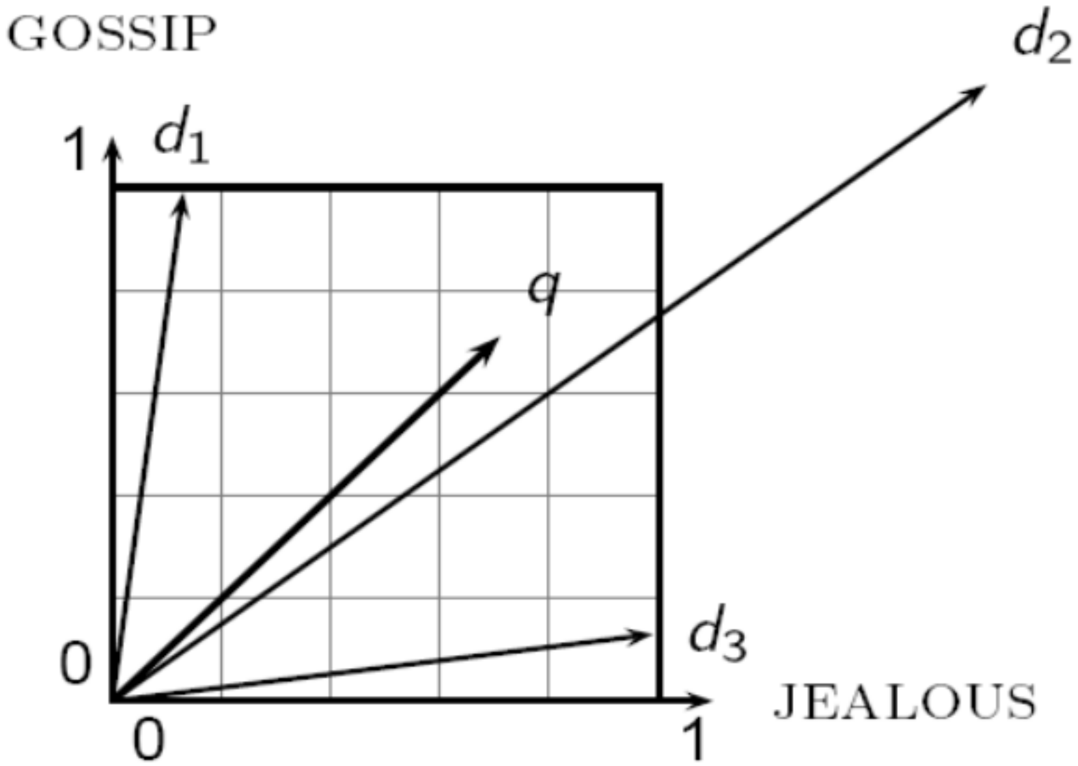
Vector space model (1)

- Treat the query as a tiny document
- Represent the query and every document each as a word vector in a word space
- Rank documents according to their proximity to the query in the word space

Vector space model (2)

Represent documents in a space of word vectors

Suppose the corpus only has two words: ‘Jealous’ and ‘Gossip’, they form a space of ‘Jealous’ and ‘Gossip’.



d1: gossip gossip jealous gossip gossip gossip gossip gossip gossip gossip gossip

d2: gossip gossip jealous gossip gossip gossip gossip gossip gossip gossip jealous jealous jealous jealous jealous jealous jealous gossip jealous

d3: jealous gossip jealous jealous jealous jealous jealous jealous jealous jealous jealous

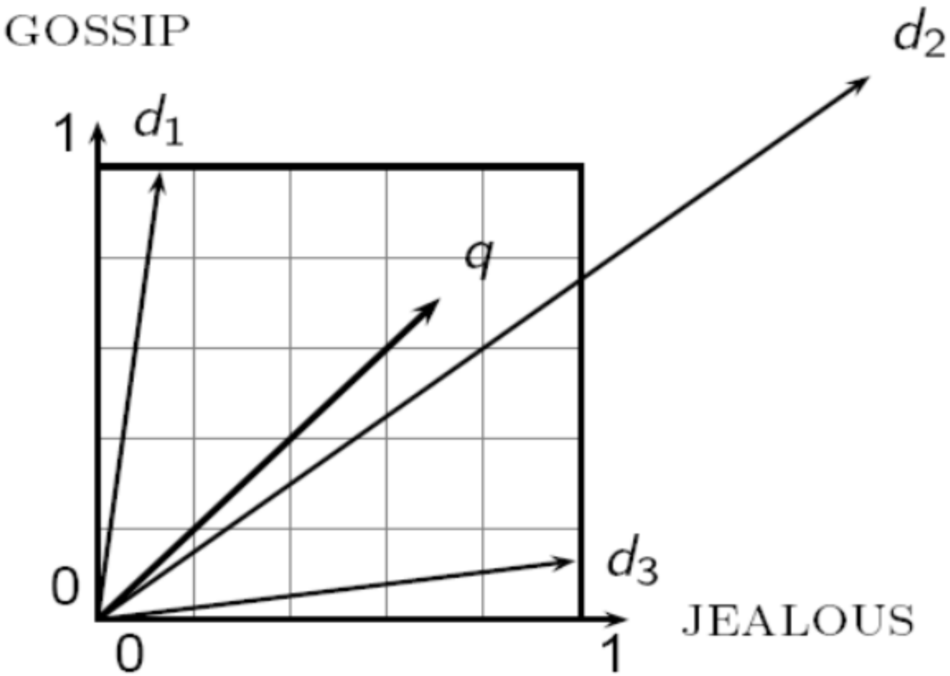
q: gossip gossip jealous gossip gossip gossip gossip gossip jealous jealous jealous jealous

Vector space model (3)

Euclidean distance

If if p = (p1, p2,..., pn) and q = (q1, q2,..., qn) are two points in the Euclidean space, their Euclidean distance is

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$



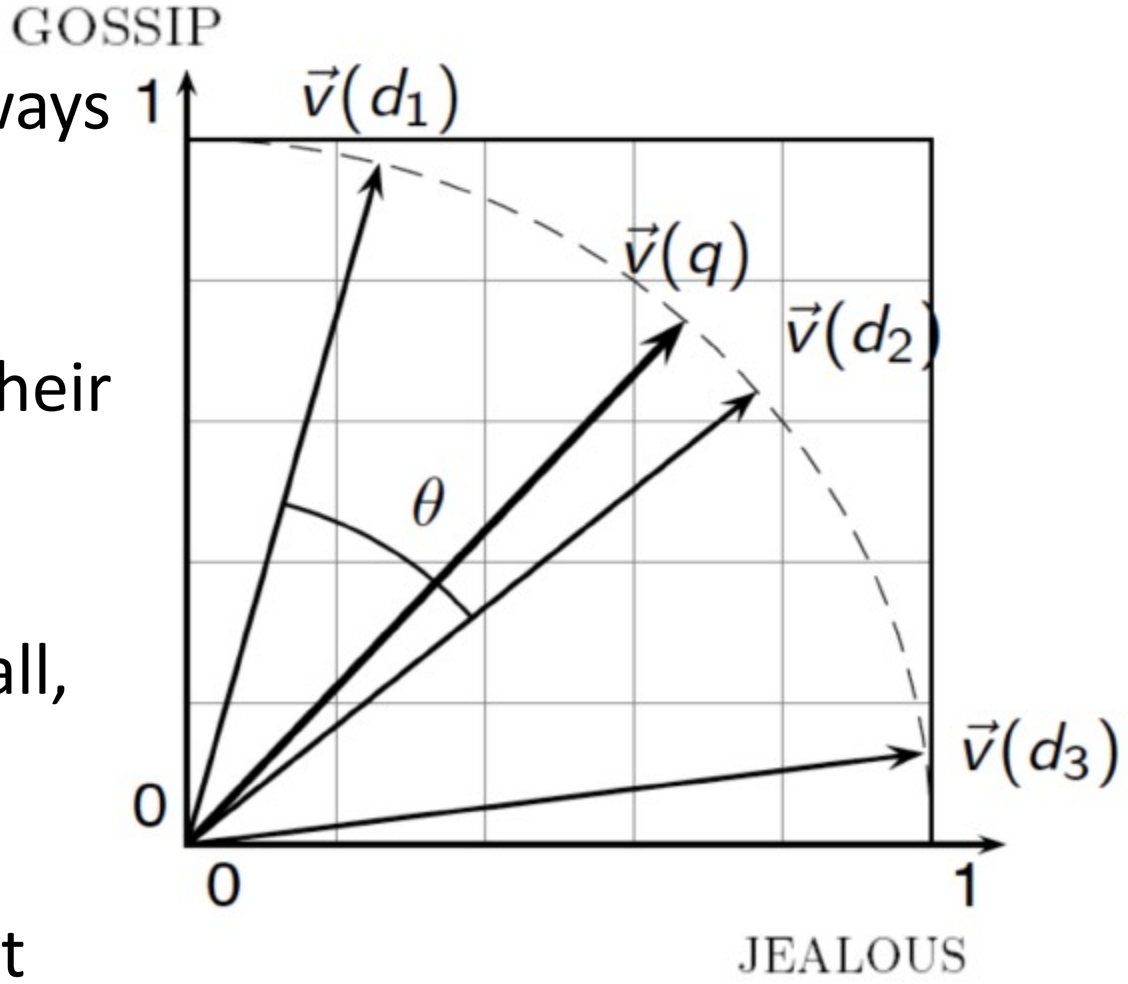
Here, if you look at the content (or we say the word distributions) of each document, d2 is actually the most similar document to q.

However, d2 produces a bigger Euclidean distance score to q.

Vector space model (4)

Use angle instead of distance

- Short query and long documents will always have big Euclidean distance
- Key idea: rank documents according to their angles with query.
- The angle between similar vectors is small, between dissimilar vectors is large.
- This is equivalent to perform a document length normalization.



Vector space model (5)

Cosine Similarity

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

$\vec{q}$ is the representation vector of the query.

$\vec{d}$ is the representation vector of the document.

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Vector space model – TF-IDF representations (1)

- Term Frequency, a measure of how frequently a term t appears in a document d:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

- $n_{t,d}$ is the number of times the term t appears in the document d.

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

Vector space model – TF-IDF representations (2)

- Inverse Document Frequency (IDF): a measure of how important a term is.

$$idf_t = log\frac{number\ of\ documents\ (N)}{number\ of\ documents\ with\ term\ t\ (df_t)}$$

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

- *IDF('this') = log(number of documents / number of documents containing the word 'this') = log(3/3) = log(1) = 0*
- *Little importance: "is", "this", "and"*
- *More importance: "scary", "long", "good"*

Vector space model – TF-IDF representations (3)

TF-IDF weighting

- Product of a term's tf weight and idf weight regarding a document

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known term weighting scheme in IR
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Ranking approaches – BM25

- It is virtually a probabilistic ranking algorithm though it looks very ad-hoc.
- For each document-query pair, compute the score functions and rank them

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

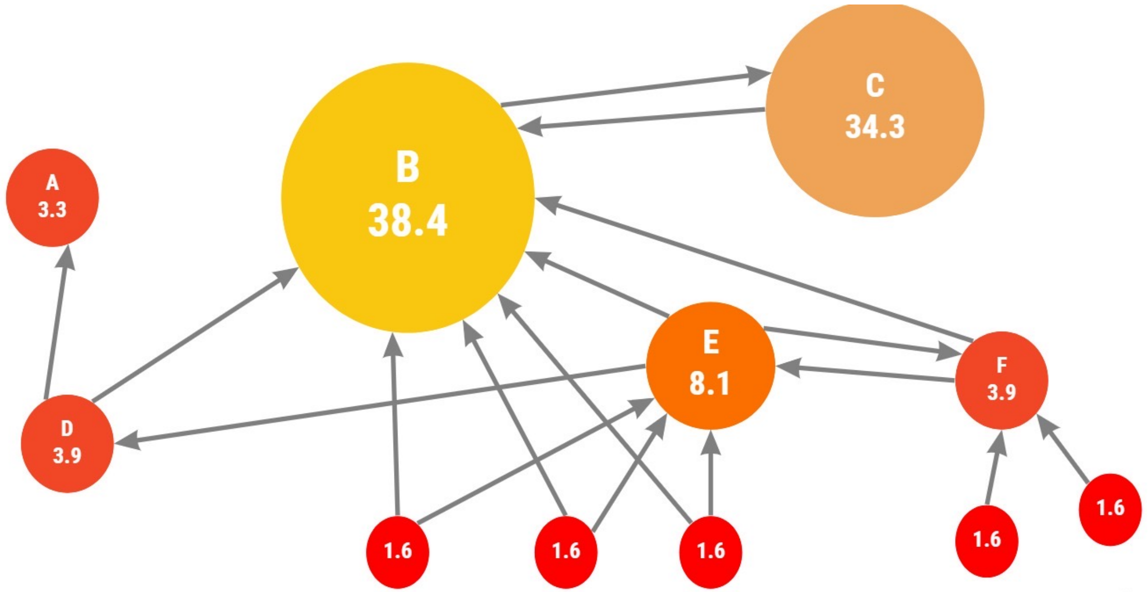
$f(q_i, D)$ is the number of times that q_i occurs in the document D, $|D|$ is the length of the document D in words,

avgdl is the average document length in the text collection. k_1 and b are free parameters.

Ranking approaches – PageRank

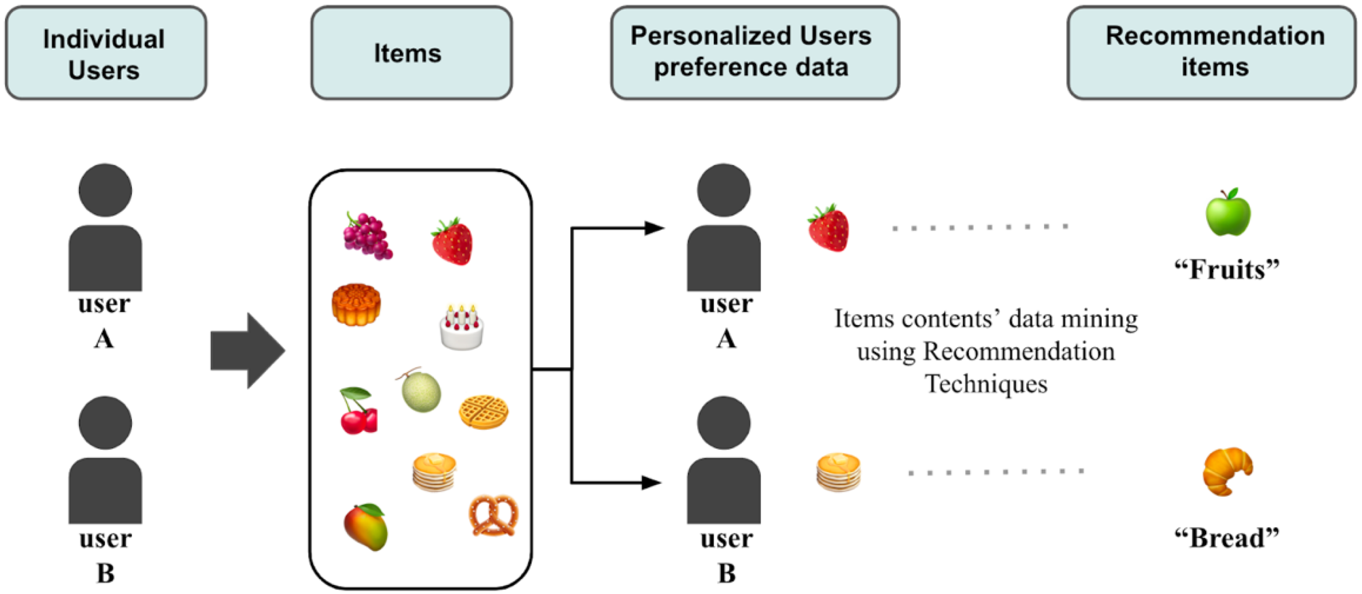
- Used by Google, to measure how important a webpage is.
- Based on page linkage: highly linked pages are more important (have greater authority) than pages with fewer links.
- Measure of query-independent importance of a page/node.
- PageRank value for a page **u** is dependent on the PageRank values for each page **v** contained in the set $\mathbf{B_u}$ (the set containing all pages linking to page **u**), divided by the number $L(v)$ of links from page **v.**

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$



Recommender systems

- Recommender systems: a subclass of information filtering system that provide suggestions for items that are most pertinent (of interest) to a particular user.

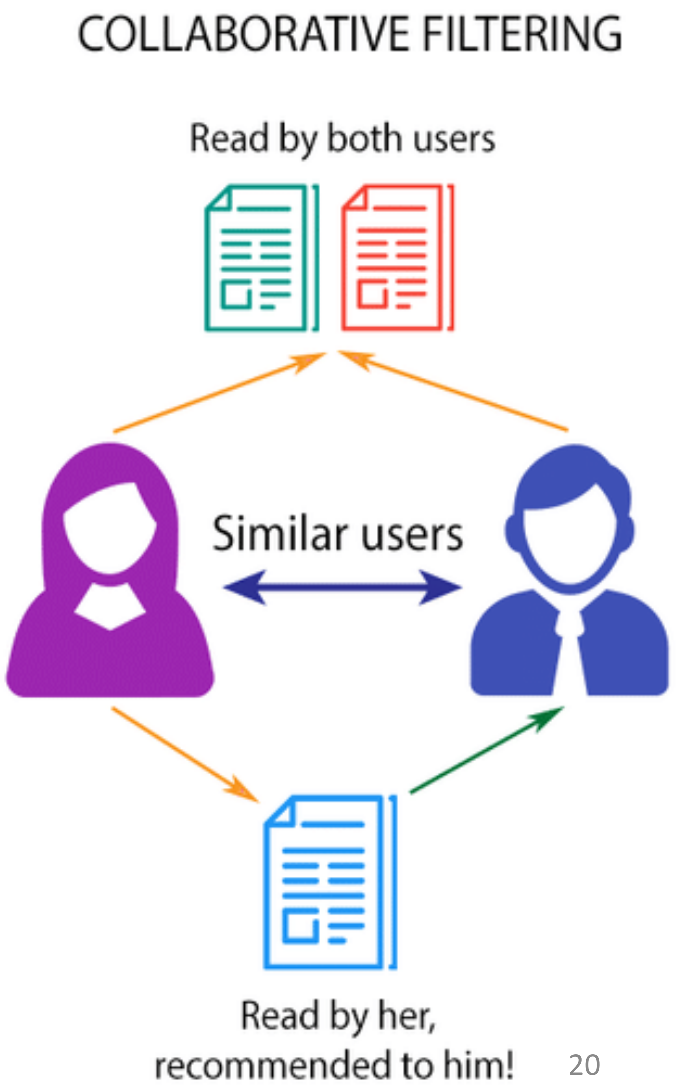


Recommendation approaches

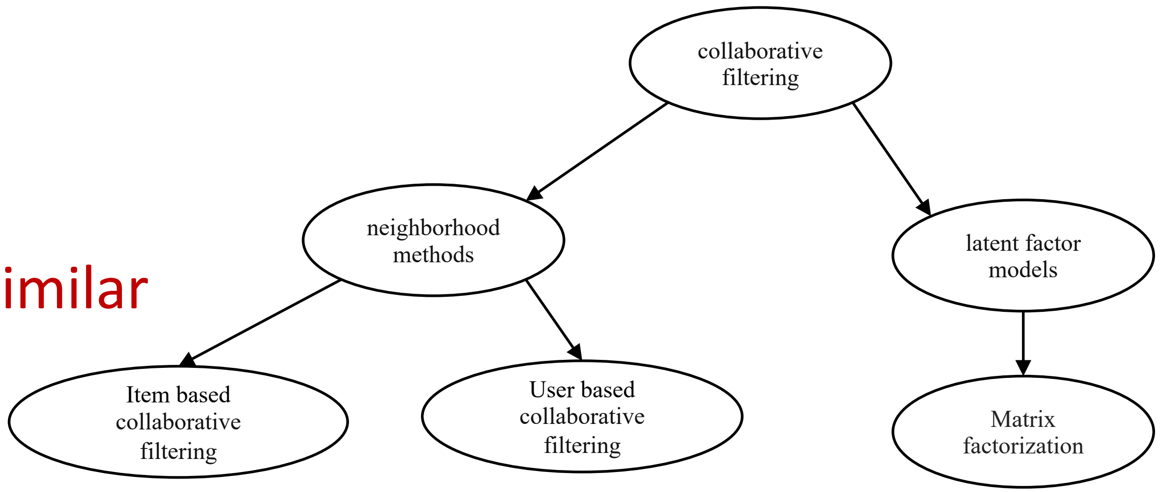
- Collaborative filtering method finds a subset of users who have similar tastes and preferences to the target user and use this subset for offering recommendations.
- Basic Assumptions :

  - Users with similar interests have common preferences.

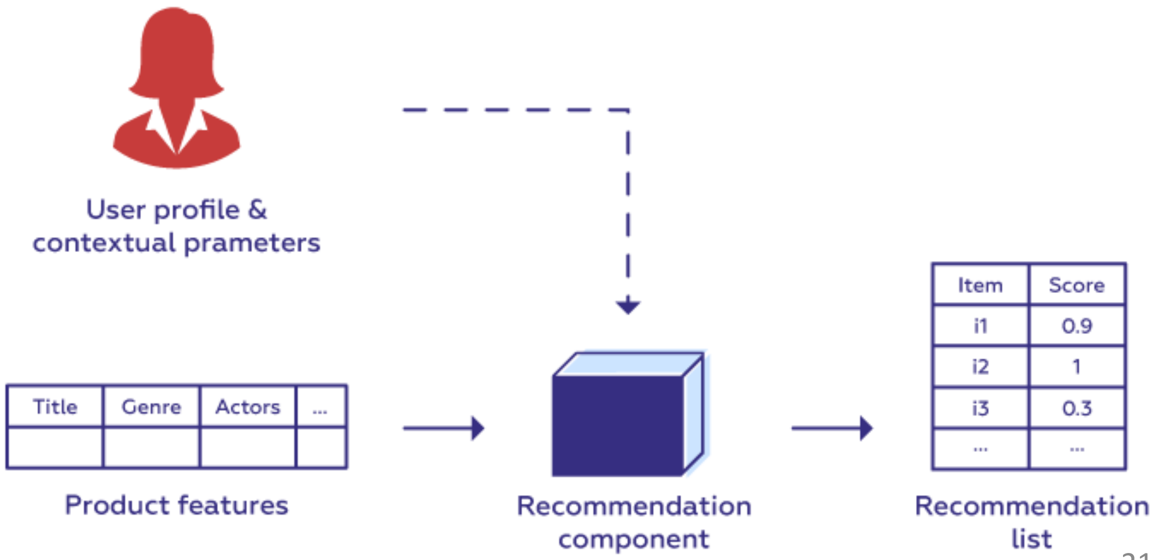
  - Sufficiently large number of user preferences are available.



Collaborative filtering



- Item/User based collaborative filtering: find similar items/users and recommend similar items.
- Matrix factorization: represent each item and user as an embedding vector and calculate their preference score (e.g., inner product).



A user/An item profile is a collection of settings and information associated with a user/an item.

Collaborative filtering methods

- 1. Weight all users with respect to similarity with the active user.
- 2. Select a subset of the users (neighbors) to use as predictors.
- 3. Normalize ratings and compute a prediction from a weighted combination of the selected neighbors' ratings.
- 4. Ranking: present items with highest predicted ratings as recommendations.

subtracting with average rating to normalize the rating scale

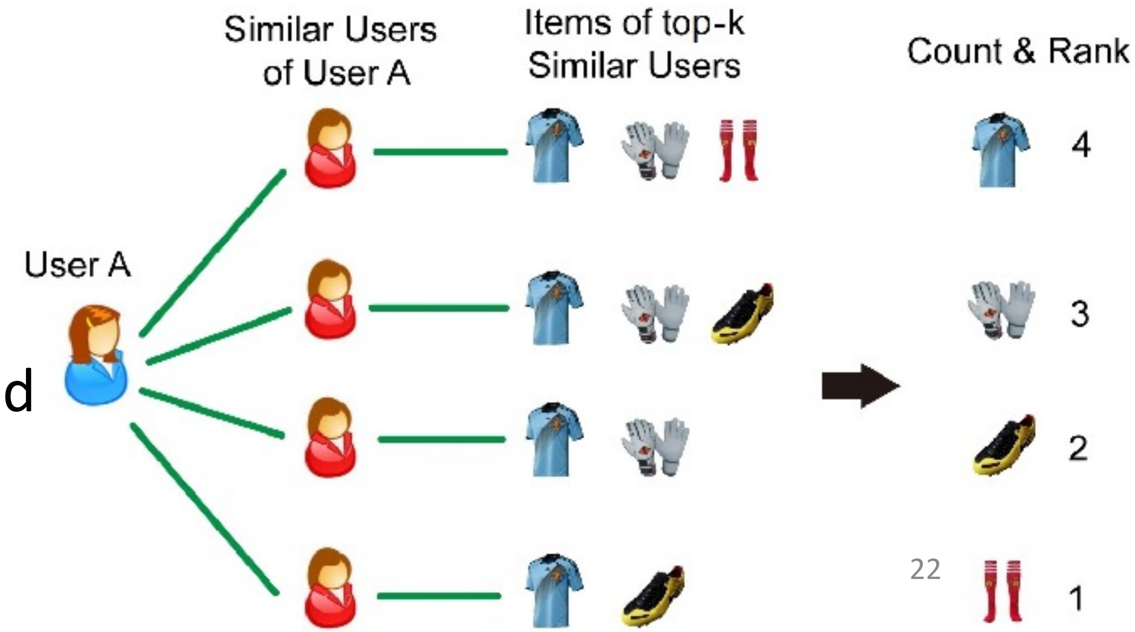
sum with user 'i's average rating

sum the multiplication of item 'i' and 'j's similarity and difference of rating by user 'u' to item 'j' and it's average rating

$$score(u,i) = \frac{\sum_j^I similarity(i,j)(r_{(u,j)} - \bar{r}_j)}{\sum_j^I similarity(i,j)} + \bar{r}_i$$

sum of item 'i' and 'j' similarity score

Make a recommendation based on user-item similarity score rankings

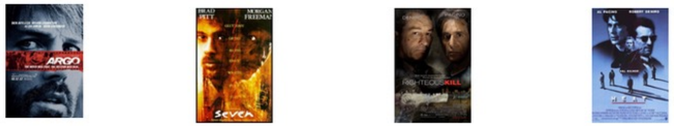


Item/User based collaborative filtering

- Calculate user-user, item-item, user-item similarity scores
  - Vector similarity (Cosine) $$Cosine(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{u \in I_u} r_{ui}^2} \sqrt{\sum_{u \in I_v} r_{vi}^2}}$$
  - Adjusted cosine vector $$ACosine(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_i})(r_{vi} - \overline{r_i})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_i})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \overline{r_i})^2}}$$
  - Pearson Correlation Coefficient (PCC) $$PCC(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \overline{r_v})^2}}$$
  - Adjusted mutual information $$MI(u, v) = \sum_{i \in I_u} \sum_{j \in I_v} p(r_{ui}, r_{vj}) \log \frac{p(r_{ui}, r_{vj})}{p(r_{ui}) p(r_{vj})}$$
  - Jaccard $$J(u, v) = \frac{|u \cap v|}{|u \cup v|}$$
  - Euclidean distance $$d(u, v) = \sqrt{\sum_{i \in I_{uv}} (r_{vi} - r_{ui})^2}$$
  - Manhattan distance $$d_1(u, v) = \sum_{i \in I_{uv}} (|r_{vi} - r_{ui}|)$$

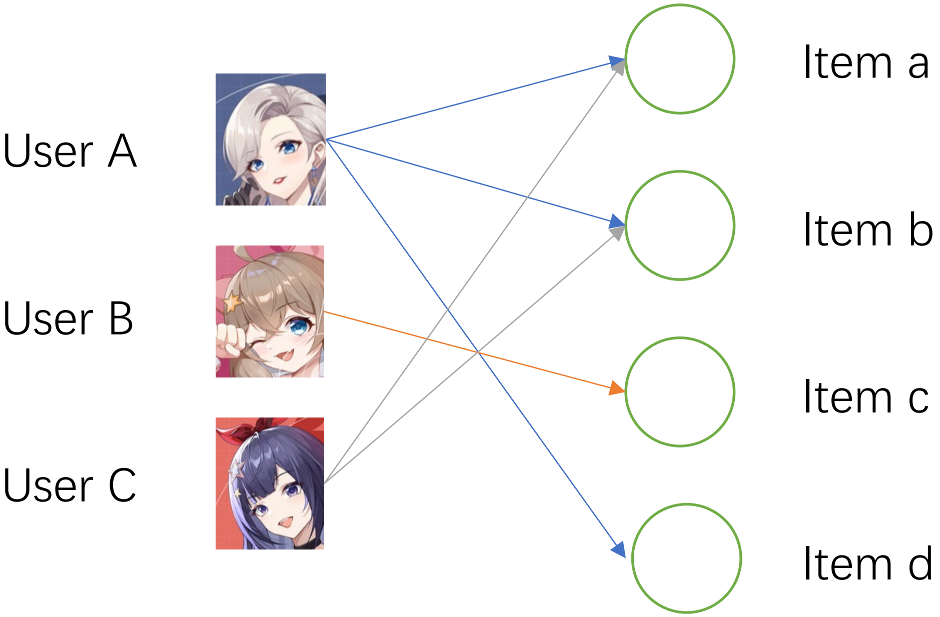
User-based collaborative filtering (1)

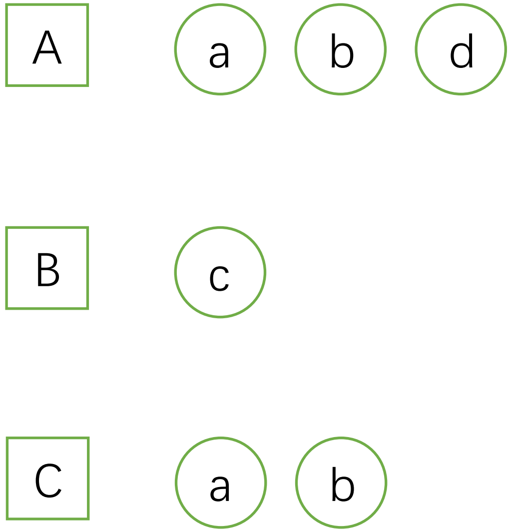
- Use user-item rating matrix
- Make user-to-user correlations
- Find highly correlated users
- Recommend items preferred by those users



John	5	1	3	5
Tom	?	?	?	2
Alice	4	?	3	?

Rating matrix





User-based collaborative filtering (2)

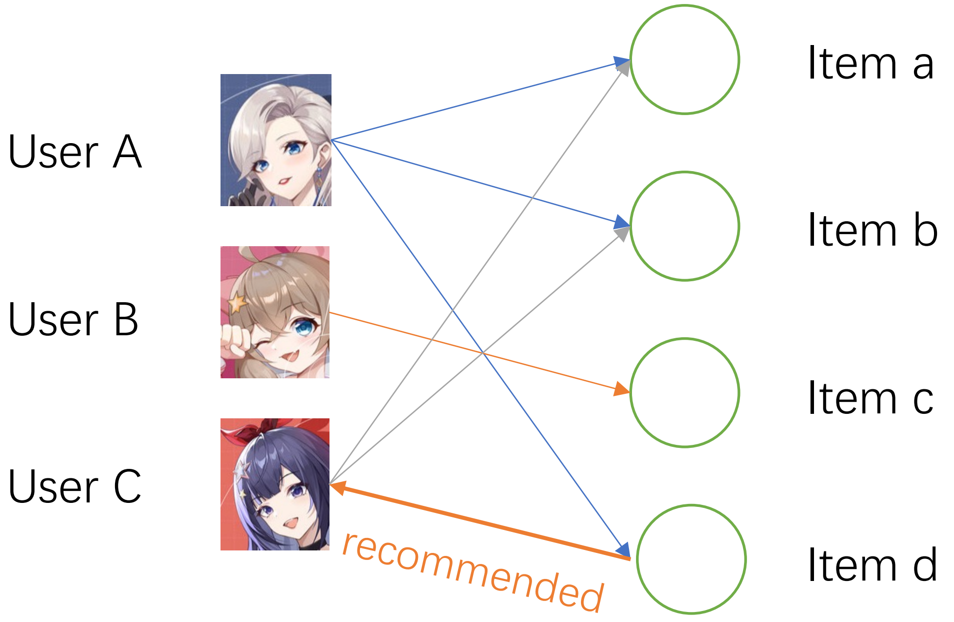
- Use user-item rating matrix
- Make user-to-user correlations
- Find highly correlated users
- Recommend items preferred by those users

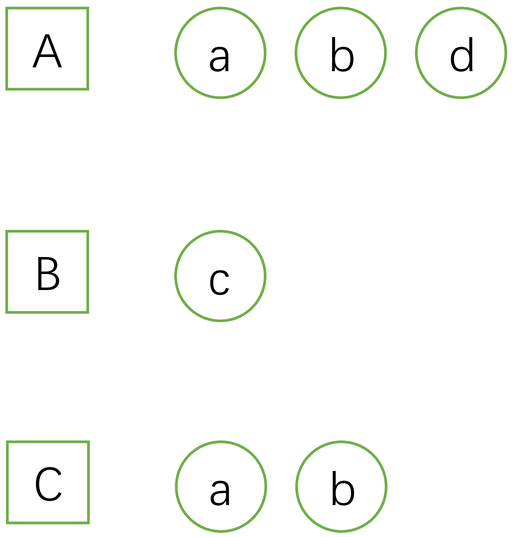
$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||_2 \times ||\vec{y}||_2} = \frac{\sum_{s\in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s\in S_{xy}} r_{x,s}^2}\sqrt{\sum_{s\in S_{xy}} r_{y,s}^2}}$$

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

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

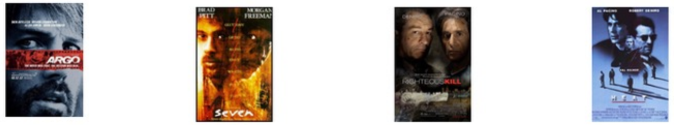
$$(c) r_{c,s} = \bar{r}_c + k\sum_{c'\in C} sim(c,c') \times (r_{c',s}, a - \bar{r}_{c'}),$$





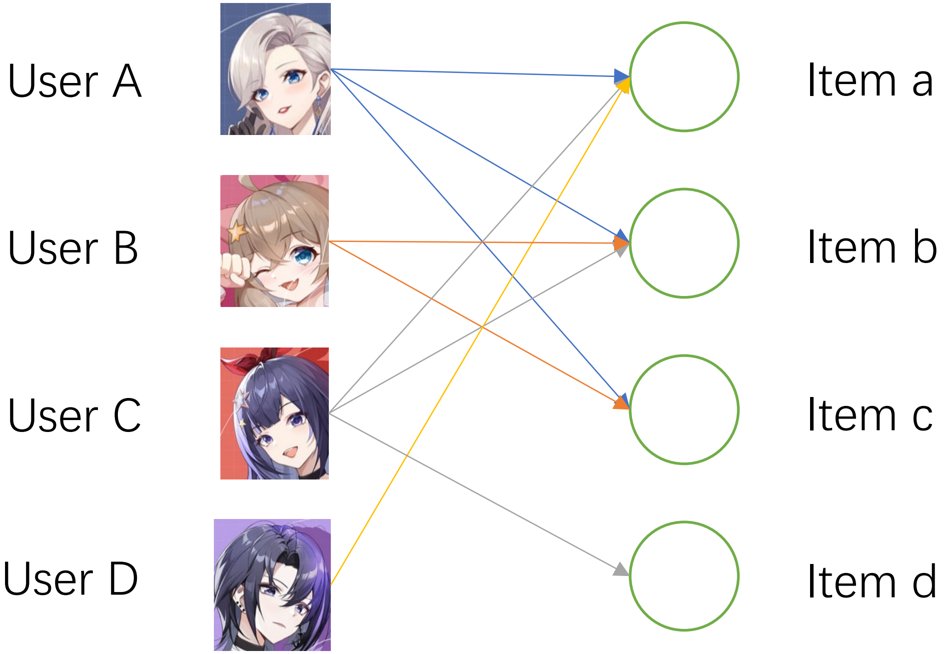
Item-based collaborative filtering (1)

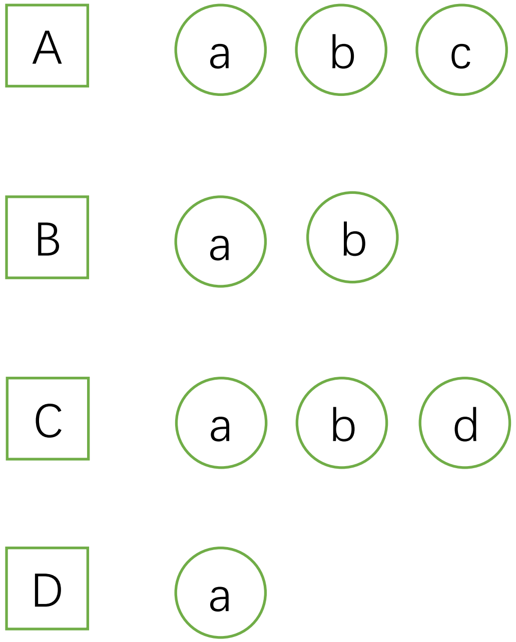
- Use user-item rating matrix
- Make item-to-item correlations
- Find highly correlated items
- Recommend user with similar items



John	5	1	3	5
Tom	?	?	?	2
Alice	4	?	3	?

Rating matrix



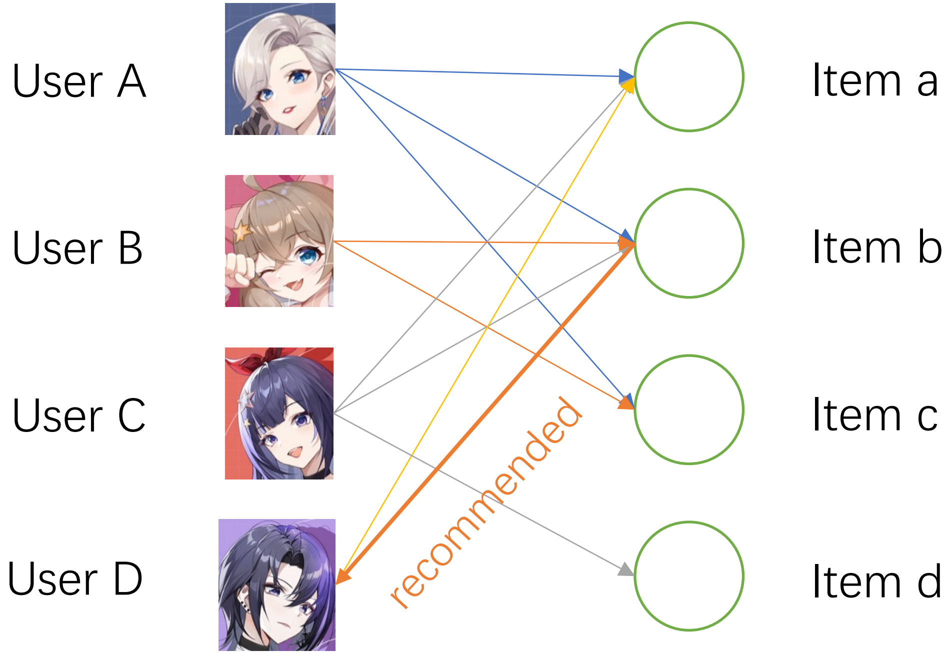


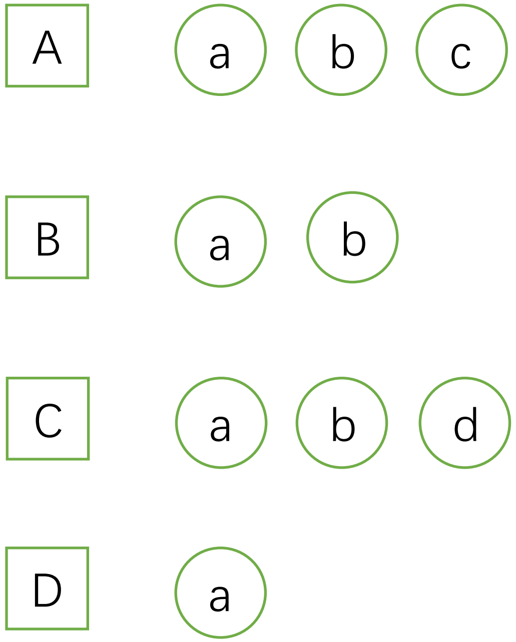
Item-based collaborative filtering (2)

- Use user-item rating matrix
- Make item-to-item correlations
- Find highly correlated items
- Recommend user with similar items

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

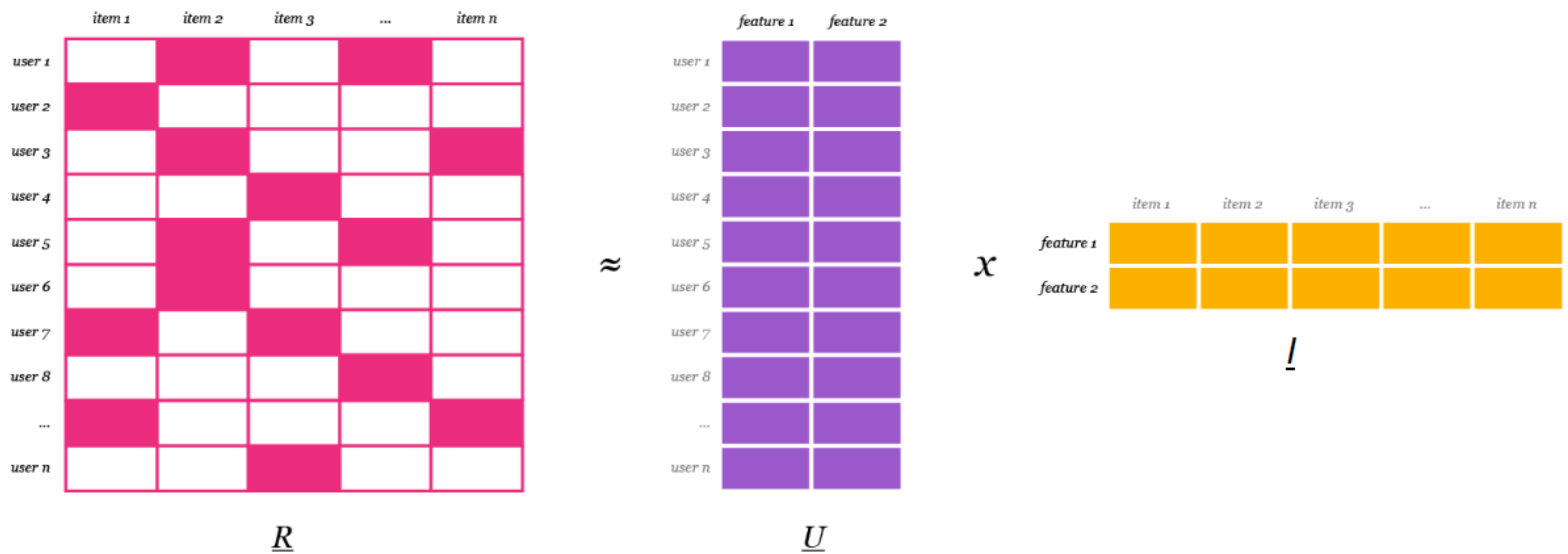
$$P_{u,i} = \frac{\sum_{all\ similar\ items,N}(s_{i,N} * U_{u,N})}{\sum_{all\ similar\ items,N}(|s_{i,N}|)}$$





Matrix factorization – illustration

- Using only user-item rating matrix (can only use profiles)
- Factorize the matrix into a multiplication of two, each row/column will represent the user/item embeddings.
- Ratings are approximated as the inner product of a user embedding and an item embedding.



Matrix factorization

- Mathematically, the problem is to approximately decompose a real (or binary) matrix $R_{m\times n}$ into a dot product of two matrices:

$$\underbrace{R_{m\times n}}_{\text{Interaction Matrix}} \approx \underbrace{P_{m\times k}}_{\text{User Matrix}} \cdot \underbrace{Q^T_{n\times k}}_{\text{Item Matrix}} .$$

- general optimization problem:

$$\min_{P,Q} \sum_{u,i\in R} \left[L(p_u, q_i, r_{ui}) + \underbrace{\gamma_p||p_u||^1 + \gamma_q||q_i||^1}_{\text{L1 Regularization}} + \underbrace{\lambda_p||p_u||^2 + \lambda_q||q_i||^2}_{\text{L2 Regularization}} \right],$$

- Loss function

$$L(p_u, q_i, r_{ui}) = \sum_{u,i\in R} \left(r_{ui} - p_u^T q_i \right)^2,$$

- model score for a user-item pair (u,i) is $p_u^T q_i = \sum_{j=1}^{k} p_{uj} \cdot q_{ij}.$

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

Neural collaborative filtering

- Input layer: using the identity of a user and an item as the input feature, transforming it to a binarized sparse vector with one-hot encoding.
- Embedding Layer: This layer turns the sparse vector into dense.
- Neural CF Layers: Multiple hidden layers to achieve nonlinear transformation
- Output Layer : Finally output a score that minimizes the pointwise between predicted and true values

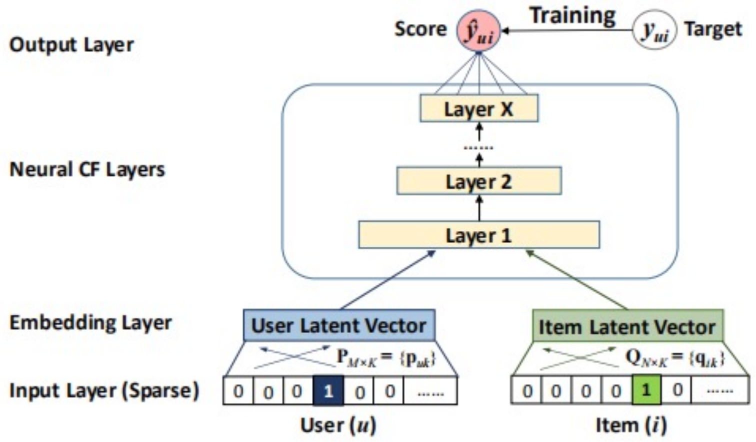


Figure 2: Neural collaborative filtering framework

$$\hat{y}_{ui} = f\left(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f\right)$$

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

$$f\left(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I\right) = \phi_{out}\left(\phi_X\left(\ldots\phi_2\left(\phi_1\left(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I\right)\right)\ldots\right)\right)$$

He, Xiangnan, et al. "Neural collaborative filtering." Proceedings of the 26th international conference on world wide web. 2017.

Neural matrix factorization

- Fusion of GMF and MLP

Latent vectors for MF and for CF.

Concatenation to calculate final scores

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

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

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

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T a\left(\mathbf{p}_u \odot \mathbf{q}_i + \mathbf{W}\begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix} + \mathbf{b}\right)\right)$$

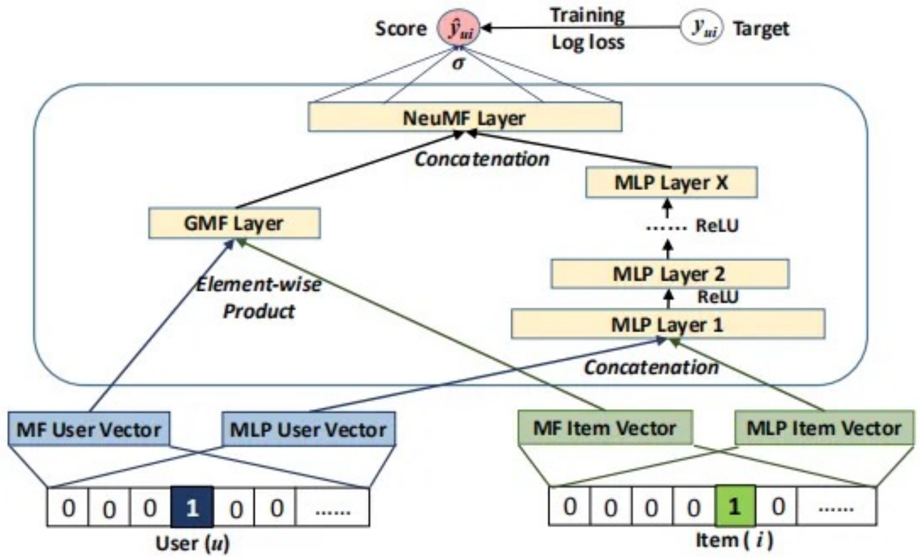


Figure 3: Neural matrix factorization model

He, Xiangnan, et al. "Neural collaborative filtering." Proceedings of the 26th international conference on world wide web. 2017.

Recommendation evaluation

- Rating prediction task

- $Root\text{-}mean\text{-}square\ error\ RMSE = \frac{\sqrt{\sum_{u,i \in T}(r_{ui} - \hat{r}_{ui})^2}}{|T|}$

- $Mean\ absolute\ error\ MAE = \frac{\sum_{u,i \in T}|r_{ui} - \hat{r}_{ui}|}{|T|}$

- Top-N recommendation task

- *Hit rate*
- $Precision = \frac{\sum_{u \in U}|R(u) \cap T(u)|}{\sum_{u \in U}|R(u)|}$, $Recall = \frac{\sum_{u \in U}|R(u) \cap T(u)|}{\sum_{u \in U}|T(u)|}$, $F_1 = \frac{2PR}{P+R}$

Recommendation challenges

- Cold start & data sparsity
- Filter bubble
- Personalization

Thanks for your attention!

Appendix

1. Introduction to Information Retrieval. C.D. Manning, P. Raghavan, H. Schütze. Cambridge UP, 2008.

2. Search Engines: Information Retrieval in Practice. W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2009.