



# CSCE 670 - Information Storage and Retrieval

## Lecture 14: Recommender Systems (Bayesian Personalized Ranking)

Yu Zhang

[yuzhang@tamu.edu](mailto:yuzhang@tamu.edu)

October 9, 2025

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F25.html>

# Implicit Feedback

- So far, we have focused mostly on estimating **explicit ratings** (e.g., 1 star – 5 stars)
- What if we only have **implicit feedback**?
  - E.g., clicks, likes, views, ...
  - Only a (small) fraction of customers who purchase a product actually leave a rating
  - Not to mention that the number of users who merely view or click on it is much larger
- **Challenge 1**: No negative feedback!
  - If I have not viewed a YouTube video, does that mean I hate it? Or I just have not been exposed to it yet?
- **Challenge 2**: Evaluation is tricky — no RMSE to measure!

# Bayesian Personalized Ranking [Rendle et al., UAI 2009]

452

RENDLE ET AL.

UAI 2009

---

## BPR: Bayesian Personalized Ranking from Implicit Feedback

---

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme

{srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de

Machine Learning Lab, University of Hildesheim

Marienburger Platz 22, 31141 Hildesheim, Germany

### Abstract

Item recommendation is the task of predicting a personalized ranking on a set of items (e.g. websites, movies, products). In this paper, we investigate the most common scenario with implicit feedback (e.g. clicks, purchases). There are many methods for item recommendation from implicit feedback like matrix factorization (MF) or adaptive k-nearest-neighbor (kNN). Even though these methods are designed for the item prediction task of personalized ranking, none of

sonalization is attractive both for content providers, who can increase sales or views, and for customers, who can find interesting content more easily. In this paper, we focus on item recommendation. The task of item recommendation is to create a user-specific ranking for a set of items. Preferences of users about items are learned from the user's past interaction with the system – e.g. his buying history, viewing history, etc.

Recommender systems are an active topic of research. Most recent work is on scenarios where users provide explicit feedback, e.g. in terms of ratings. Nevertheless, in real-world scenarios most feedback is not explicit but implicit. Implicit feedback is tracked au-

# Big Idea: Let's go back to ranking!

- Our previous discussion has focused on ratings prediction (e.g., optimizing RMSE)
- But with implicit feedback, let's instead focus on how well we rank items
- Example of implicit feedback



*Exile on Main St.*



*Run the Jewels*



*Pet Sounds*



*Illmatic*



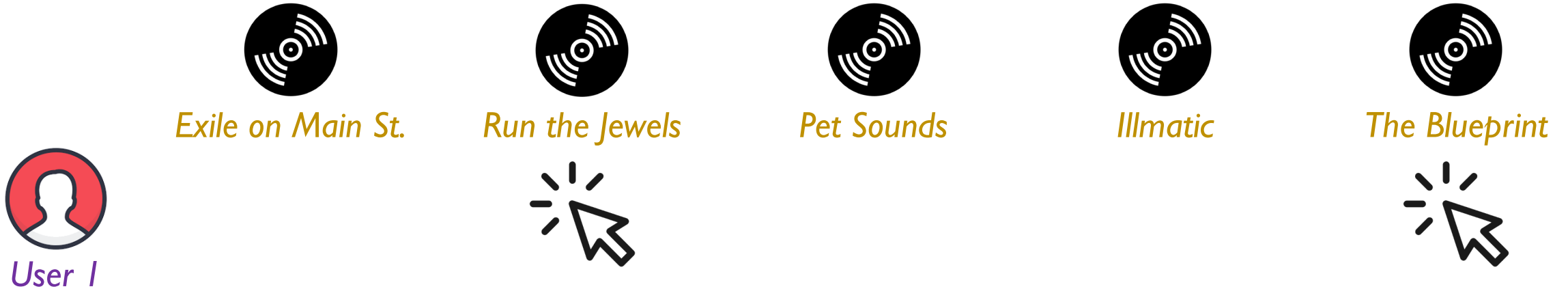
*The Blueprint*



*User 1*



# From Implicit Feedback to Item Ranking



- Can we derive the **pairwise** preferences for *User 1*?

	<i>Exile on Main St.</i>	<i>Run the Jewels</i>	<i>Pet Sounds</i>	<i>Illmatic</i>	<i>The Blueprint</i>
<i>Exile on Main St.</i>					
<i>Run the Jewels</i>					
<i>Pet Sounds</i>					
<i>Illmatic</i>					
<i>The Blueprint</i>					

# From Implicit Feedback to Item Ranking

+ = user prefers the column item  
 - = user prefers the row item  
 ? = we don't know

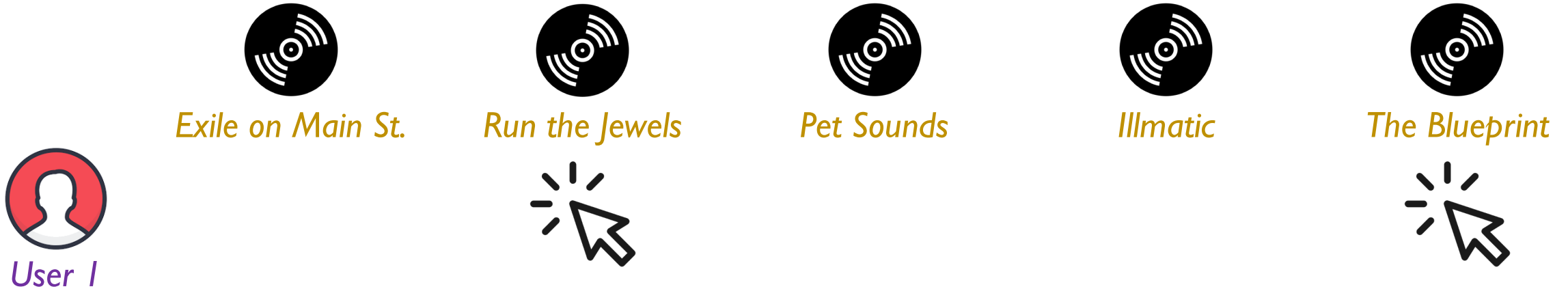


- **Principle 1:** The user prefers each clicked item over all other non-observed items.

	<i>Exile on Main St.</i>	<i>Run the Jewels</i>	<i>Pet Sounds</i>	<i>Illmatic</i>	<i>The Blueprint</i>
<i>Exile on Main St.</i>		+			+
<i>Run the Jewels</i>	-		-	-	
<i>Pet Sounds</i>		+			+
<i>Illmatic</i>		+			+
<i>The Blueprint</i>	-		-	-	

# From Implicit Feedback to Item Ranking

+ = user prefers the column item  
 - = user prefers the row item  
 ? = we don't know

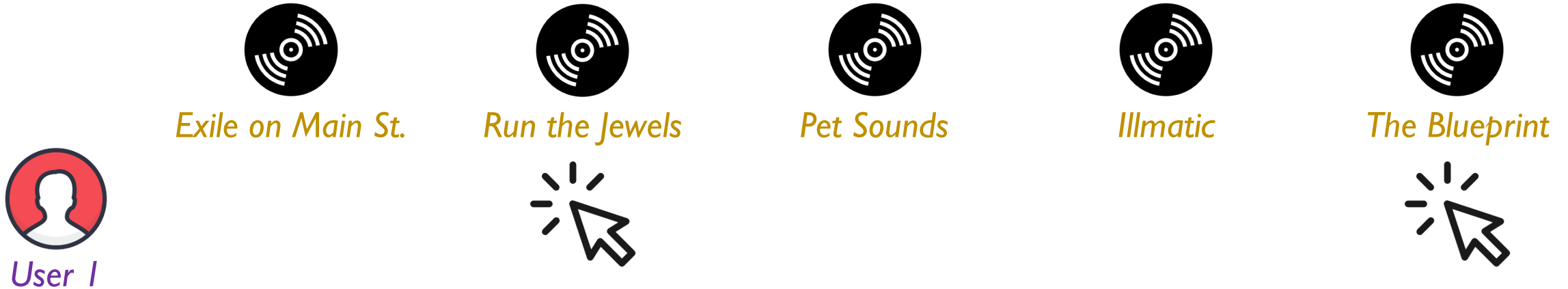


- Principle 2:** We cannot infer any preference between two non-observed items.

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.		+	?	?	+
Run the Jewels	-		-	-	
Pet Sounds	?	+		?	+
Illmatic	?	+	?		+
The Blueprint	-		-	-	

# From Implicit Feedback to Item Ranking

+ = user prefers the column item  
 - = user prefers the row item  
 ? = we don't know



- Principle 3:** We cannot infer any preference between two clicked items.

	Exile on Main St.	Run the Jewels	Pet Sounds	Illmatic	The Blueprint
Exile on Main St.		+	?	?	+
Run the Jewels	-		-	-	?
Pet Sounds	?	+		?	+
Illmatic	?	+	?		+
The Blueprint	-	?	-	-	



# Consider All Users

- From one **user-item matrix**
  - + = clicked
  - ? = not observed
- To many **item-item matrices**
  - + = the user prefers the column item
  - = the user prefers the row item
  - ? = we don't know

	$i_1$	$i_2$	$i_3$	$i_4$
$u_1$	?	+	+	?
$u_2$	+	?	?	+
$u_3$	+	+	?	?
$u_4$	?	?	+	+
$u_5$	?	?	+	?

← item →

↑ user ↓

$u_1: i >_{u_1} j$				
	$i_1$	$i_2$	$i_3$	$i_4$
$j_1$		+	+	?
$j_2$	-		?	-
$j_3$	-	?		-
$j_4$	?	+	+	

← item →

↑ item ↓

...

$u_5: i >_{u_5} j$				
	$i_1$	$i_2$	$i_3$	$i_4$
$j_1$		?	+	?
$j_2$	?		+	?
$j_3$	-	-		-
$j_4$	?	?	+	

← item →

↑ item ↓

# Our Task

- To predict the **personalized preference** between any two items
  - Given a user  $u$  and two items  $i \neq j$ , is it
    - $i >_u j$ , or
    - $i <_u j$ ?
- Equivalently, to estimate a **personalized ranking function**  $x(u, i, j)$ :
  - $x(u, i, j) > 0$  if  $i$  is preferred by  $u$
  - $x(u, i, j) < 0$  if  $j$  is preferred by  $u$
- How can we know if we are doing a good job?
  - **Challenge 2:** Evaluation is tricky — no RMSE to measure!

# Counting the Successes

- Let's count all the times the function correctly guesses.
- Define  $\delta(\hat{x}(u, i, j) > 0) = \begin{cases} 1, & \text{if } x(u, i, j) > 0, \\ 0, & \text{otherwise.} \end{cases}$ 
  - If the ground truth is  $i >_u j$ , and you predict  $i >_u j$ , you get a score of 1.
  - If the ground truth is  $i >_u j$ , and you predict  $i <_u j$ , you get a score of 0.
- Enumerate all the cases where  $i >_u j$ :

$$\text{AUC}(u) = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

$I_u^+$ : the set of items clicked by  $u$

“Area Under the Curve” (AUC)

# AUC

$$\text{AUC}(u) = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

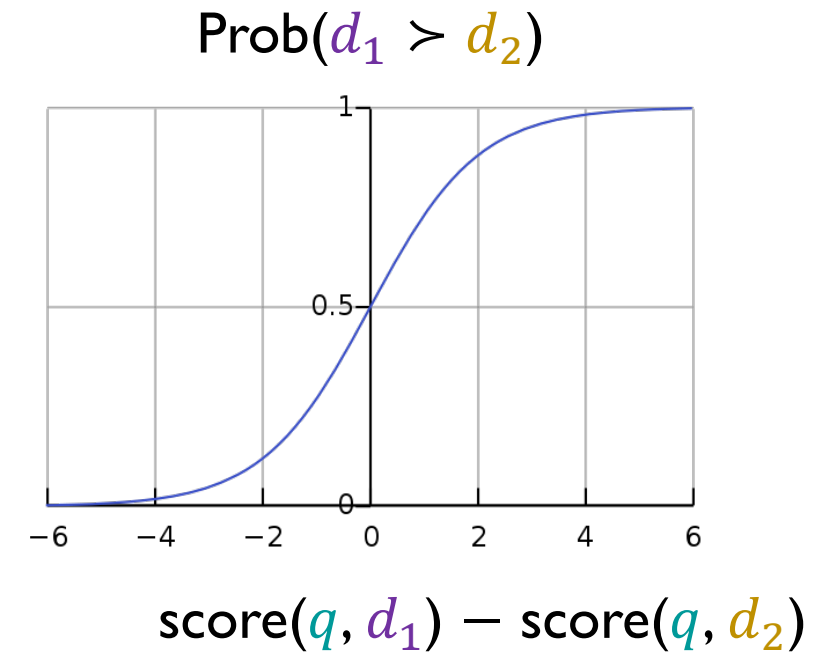
- AUC essentially counts how many times the model correctly identifies that  $u$  prefers the item they clicked (positive feedback) over the item they did not.
- What is the AUC of a perfect model?
  - 1
- What is the AUC of a random guesser?
  - 0.5
- Then we can find the average AUC for all users

$$\text{AUC} = \frac{1}{|U|} \sum_{u \in U} \text{AUC}(u)$$

# AUC

$$\text{AUC}(u) = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

- But how to develop a model to calculate  $\hat{x}(u, i, j)$ ?
- Recall pairwise learning to rank (e.g., RankNet)
  - Given a query  $q$  and two documents  $d_1$  and  $d_2$
  - **Step 1:** We calculate  $\text{score}(q, d_1)$  and  $\text{score}(q, d_2)$
  - **Step 2:** We calculate their difference  $\text{score}(q, d_1) - \text{score}(q, d_2)$
  - **Step 3:** Based on this difference and the ground truth, we train the model using gradient descent.



# Extending the Idea

- Given a user  $u$  and two items  $i$  and  $j$ , we want to calculate  $\hat{x}(u, i, j)$

$$\hat{x}(u, i, j) = \hat{x}(u, i) - \hat{x}(u, j)$$

- $\hat{x}(u, i)$  larger if  $u$  likes  $i$  to a greater extent
- $\hat{x}(u, i)$  smaller if  $u$  likes  $i$  to a lesser extent
- Independent of  $\hat{x}(u, j)$

# Example

- Suppose
  - $\hat{x}(u, \text{Exile on Main St.}) = 2$
  - $\hat{x}(u, \text{Run the Jewels}) = 8$
  - $\hat{x}(u, \text{Illmatic}) = 3$
  - $\hat{x}(u, \text{The Blueprint}) = 7$
- What is  $\text{AUC}(u)$ ?

		2	8	3	7
		<i>Exile on Main St.</i>	<i>Run the Jewels</i>	<i>Illmatic</i>	<i>The Blueprint</i>
2	<i>Exile on Main St.</i>		+	?	+
8	<i>Run the Jewels</i>	-		-	?
3	<i>Illmatic</i>	?	+		+
7	<i>The Blueprint</i>	-	?	-	

# Example

		2	8	3	7
		<i>Exile on Main St.</i>	<i>Run the Jewels</i>	<i>Illmatic</i>	<i>The Blueprint</i>
2	<i>Exile on Main St.</i>		+	?	+
8	<i>Run the Jewels</i>	-		-	?
3	<i>Illmatic</i>	?	+		+
7	<i>The Blueprint</i>	-	?	-	

- All 8 entries are correctly predicted, so  $AUC(u) = \frac{1}{8} \times 8 = 1$

		3	7	2	8
		<i>Exile on Main St.</i>	<i>Run the Jewels</i>	<i>Illmatic</i>	<i>The Blueprint</i>
3	<i>Exile on Main St.</i>		+	?	+
7	<i>Run the Jewels</i>	-		-	?
2	<i>Illmatic</i>	?	+		+
8	<i>The Blueprint</i>	-	?	-	



# Example

- What are the rankings in the previous two examples?
  - [*Run the Jewels*, *The Blueprint*, *Illmatic*, *Exile on Main St.*]
  - [*The Blueprint*, *Run the Jewels*, *Exile on Main St.*, *Illmatic*,]
  - Both are [Clicked, Clicked, Not Clicked, Not Clicked]
- What is the AUC if we have [Clicked, Not Clicked, Clicked, Not Clicked]?
  - $\text{AUC} = \frac{1}{4}(1 + 1 + 0 + 1) = 0.75$

How to calculate  $\hat{x}(u, i)$ ?

# One Idea: Matrix Factorization!

- $\hat{x}(u, i) = \mathbf{q}_i \mathbf{p}_u^T$
- $\hat{x}(u, i, j) = \hat{x}(u, i) - \hat{x}(u, j) = \mathbf{q}_i \mathbf{p}_u^T - \mathbf{q}_j \mathbf{p}_u^T = (\mathbf{q}_i - \mathbf{q}_j) \mathbf{p}_u^T$
- **Task:** Learning all user vectors  $\mathbf{p}_x^T$  and item vectors  $\mathbf{q}_i$  from implicit feedback
- Did we see anything similar (but not quite the same) before?

The diagram illustrates matrix factorization. On the left, a 6x12 matrix of user-item ratings is shown, with some cells highlighted in yellow. This matrix is approximated (indicated by  $\approx$ ) by the product of two matrices: a 6x3 matrix of user factors and a 3x12 matrix of item factors. The user factors matrix is labeled 'users' and the item factors matrix is labeled 'items'.

items \ users	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		5			5		4		
2		4		1	2		3		4	3	5	
3			4		5			4			2	
4			4	3	4	2				2	5	
5	1		3		3		2			4		
6												

users	1	2	3
1	.1	-.4	.2
2	-.5	.6	.5
3	-.2	.3	.5
4	1.1	2.1	.3
5	-.7	2.1	-.2
6	-.1	.7	.3

items	1	2	3	4	5	6	7	8	9	10	11	12
1	1.1	-.2	.3	.5	-.2	-.5	.8	-.4	.3	1.4	2.4	-.9
2	-.8	.7	.5	1.4	.3	-.1	1.4	2.9	-.7	1.2	-.1	1.3
3	2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

## Slide 32 in Lecture 9: Training a Linear SVM for Ranking

- But we don't have pointwise training data!
  - Remember we only have lots of  $(c_i, c_k)$ , where we already know  $f(\psi_i) > f(\psi_k)$
  - We don't know the value of  $f(\psi_i)$  or  $f(\psi_k)$
- **Idea:** Create a new instance space from pairwise learning
  - We have  $c_i > c_k \iff f(\psi_i) > f(\psi_k)$
  - We also have  $f(\psi_i) = w^T \psi_i$  and  $f(\psi_k) = w^T \psi_k$
  - So  $c_i > c_k \iff w^T \psi_i > w^T \psi_k \iff w^T (\psi_i - \psi_k) > 0$
  - Let's create a new instance  $\phi_u = \psi_i - \psi_k$
  - And  $z_u = +1, 0, -1$  as  $c_i >, =, < c_k$
  - From training data  $\mathcal{S} = \{\phi_u\}$ , we train an SVM

# Learning Objective

- **Task:** Learning all user vectors  $\mathbf{p}_x^T$  and item vectors  $\mathbf{q}_i$  from implicit feedback
- What is our learning objective?
  - When we have **explicit** feedback:
  - **Evaluation metric:** RMSE
  - **Learning objective:** RMSE
  - When we have **implicit** feedback:

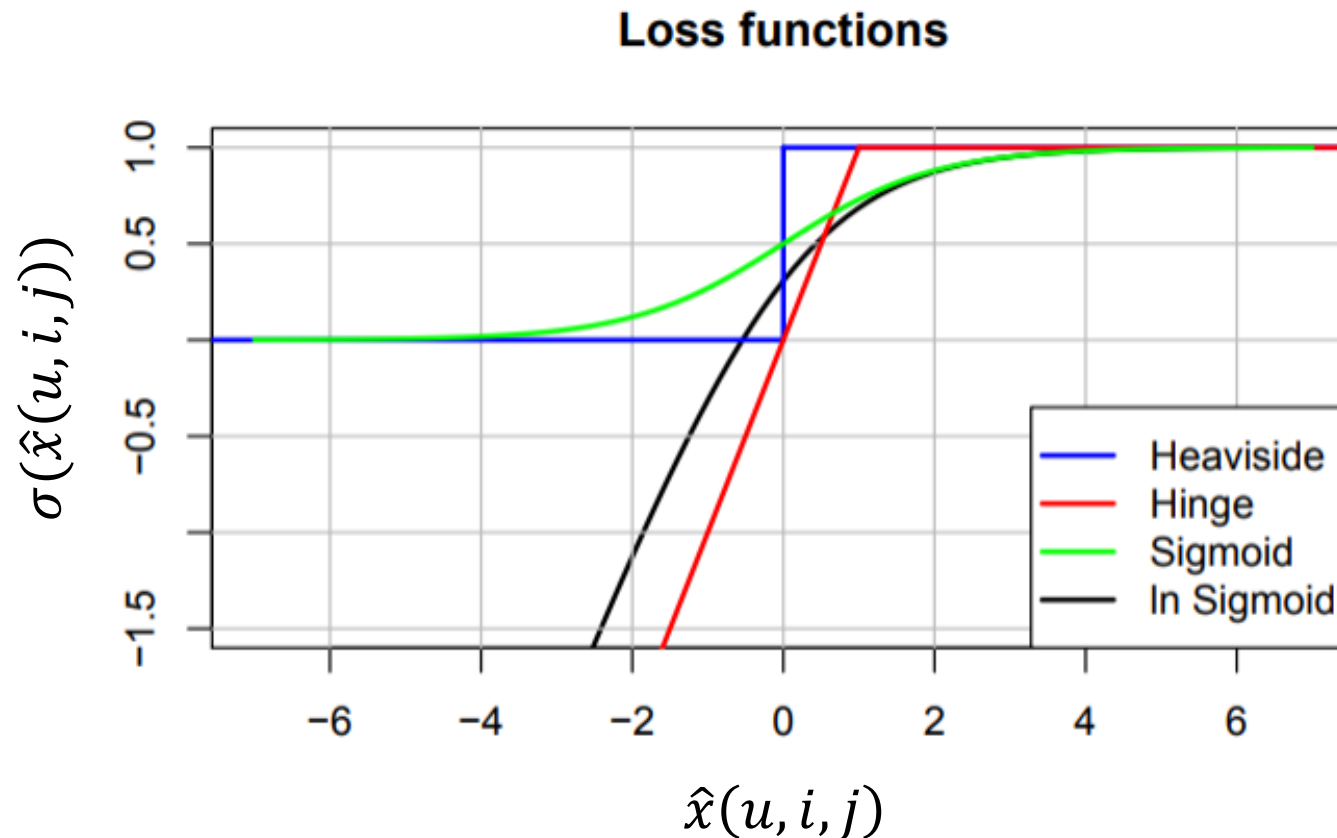
- **Evaluation metric:** AUC

$$\text{AUC}(u) = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \delta(\hat{x}(u, i, j) > 0)$$

- **Learning objective:** Can we use AUC?
  - No! Because  $\delta(\cdot)$  is not differentiable.

# Can we replace $\delta(\hat{x}(u, i, j) > 0)$ with something similar?

- Instead of a counting function  $\delta(\hat{x}(u, i, j) > 0)$  (1 or 0), let's pick a smooth function  $\sigma(\hat{x}(u, i, j))$



# BPR uses Log Sigmoid

$$\sigma(x) = \ln \frac{1}{1 + e^{-x}}$$

- Then the learning objective becomes:

$$J = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \sigma \left( \boxed{(\mathbf{q}_i - \mathbf{q}_j) \mathbf{p}_u^T} \right) \hat{x}(u, i, j)$$

- Gradient Descent

$$\frac{\partial J}{\partial \theta} = \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \frac{d\sigma}{d\hat{x}} \cdot \frac{\partial \hat{x}}{\partial \theta}$$

- What is  $\frac{d\sigma}{d\hat{x}}$  ?

- $\frac{d\sigma}{d\hat{x}} = \frac{e^{-\hat{x}}}{1 + e^{-\hat{x}}}$

# BPR uses Log Sigmoid

- Then the learning objective becomes:

$$J = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \sigma \left( \overbrace{(\mathbf{q}_i - \mathbf{q}_j) \mathbf{p}_u^T}^{\hat{x}(u, i, j)} \right)$$

- Gradient Descent

$$\frac{\partial J}{\partial \theta} = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \frac{d\sigma}{d\hat{x}} \cdot \frac{\partial \hat{x}}{\partial \theta}$$

- What is  $\frac{\partial \hat{x}}{\partial \theta}$  ?

- If  $\theta$  is  $\mathbf{q}_i$ ,  $\frac{\partial \hat{x}}{\partial \theta} = \mathbf{p}_u^T$
- If  $\theta$  is  $\mathbf{q}_j$ ,  $\frac{\partial \hat{x}}{\partial \theta} = -\mathbf{p}_u^T$
- If  $\theta$  is  $\mathbf{p}_u^T$ ,  $\frac{\partial \hat{x}}{\partial \theta} = \mathbf{q}_i - \mathbf{q}_j$



# The Gradient Descent Process

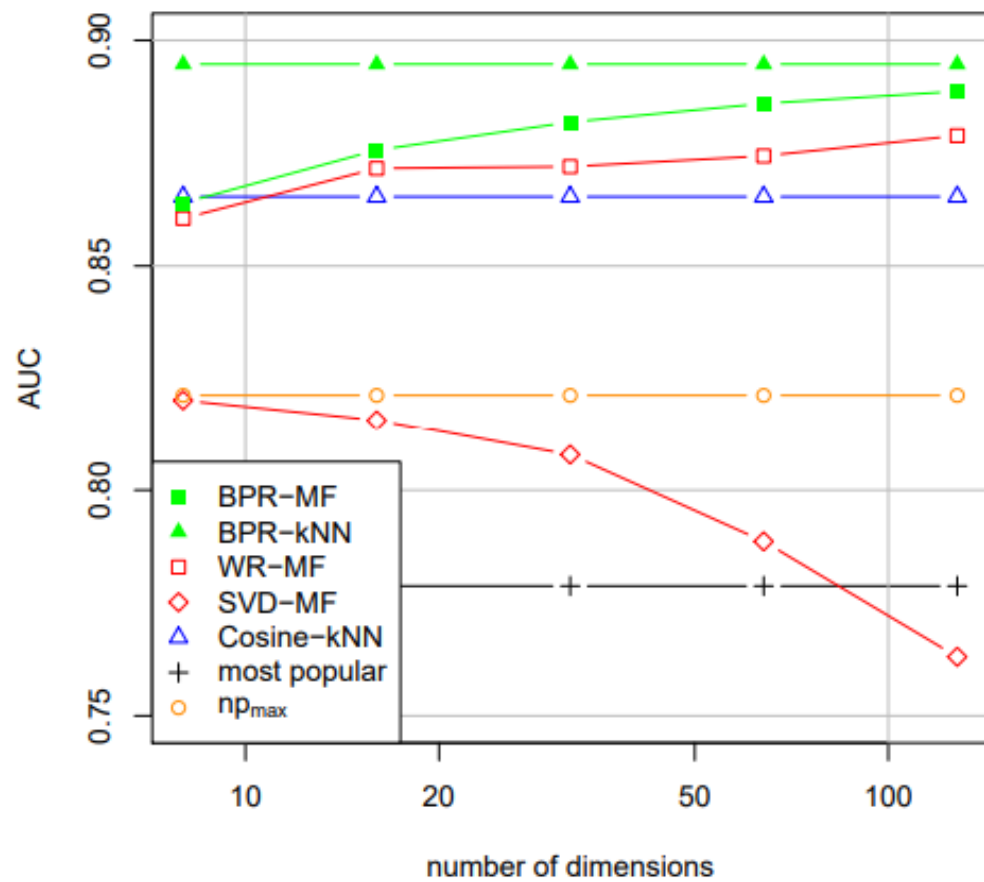
```
1: procedure LEARNBPR( $D_S, \Theta$ )
2:   initialize  $\Theta$ 
3:   repeat
4:     draw  $(u, i, j)$  from  $D_S$ 
5:      $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_{\Theta} \cdot \Theta \right)$ 
6:   until convergence
7:   return  $\hat{\Theta}$ 
8: end procedure
```

Regularization:  
 $\lambda_{\Theta} \|\Theta\|^2$  in the objective

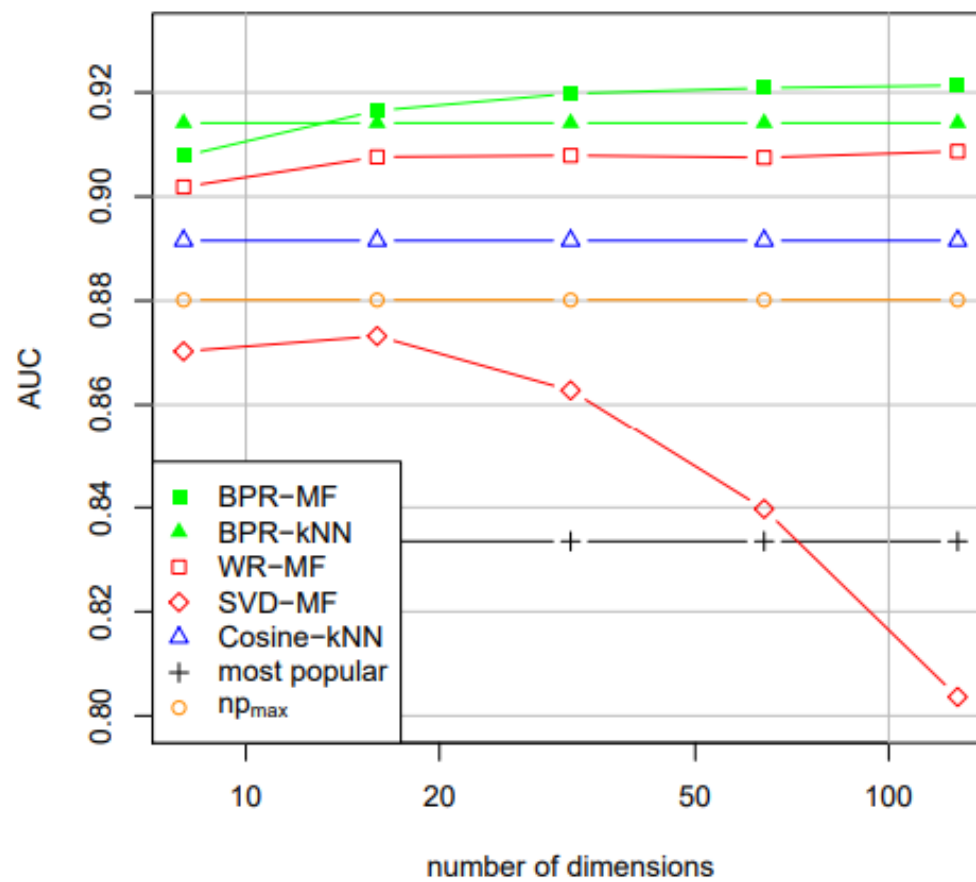
Figure 4: Optimizing models for BPR with bootstrapping based stochastic gradient descent. With learning rate  $\alpha$  and regularization  $\lambda_{\Theta}$ .

# Does BPR really work?

Online shopping: Rossmann



Video Rental: Netflix



# Takeaways

- BPR is a foundational method for recommendation with implicit feedback.
- BPR optimizes for ranking not for ratings.
- We can build BPR on top of other recommendation methods, but a classic approach uses Matrix Factorization.

## Going Back to Search Engines

# Can RecSys techniques inspire us to improve our search engine?

- What problems does our lexical-based search engine (TF-IDF and BM25) have now?
- Query: “*large language models*”
- TF-IDF and BM25 can only find documents containing “*large*” or “*language*” or “*models*”
- How about the following documents?
  - “*ChatGPT is a generative artificial intelligence chatbot developed by OpenAI and released in 2022.*”
  - “*In-the-Flow Agentic System Optimization for Effective Planning and Tool Use*”
- Some words are semantically close, but not lexically the same (or similar).
  - E.g., “*car*” vs. “*automobile*”; “*agricultural*” vs. “*farming*”
- Some documents are semantically close, but do not share any common words.

# Document-Word Occurrence Matrix

- Given a document  $d$  and a word  $w$ , let's define

$$U_{dw} = \begin{cases} 1, & \text{if } w \text{ appears in } d \\ 0, & \text{otherwise} \end{cases}$$

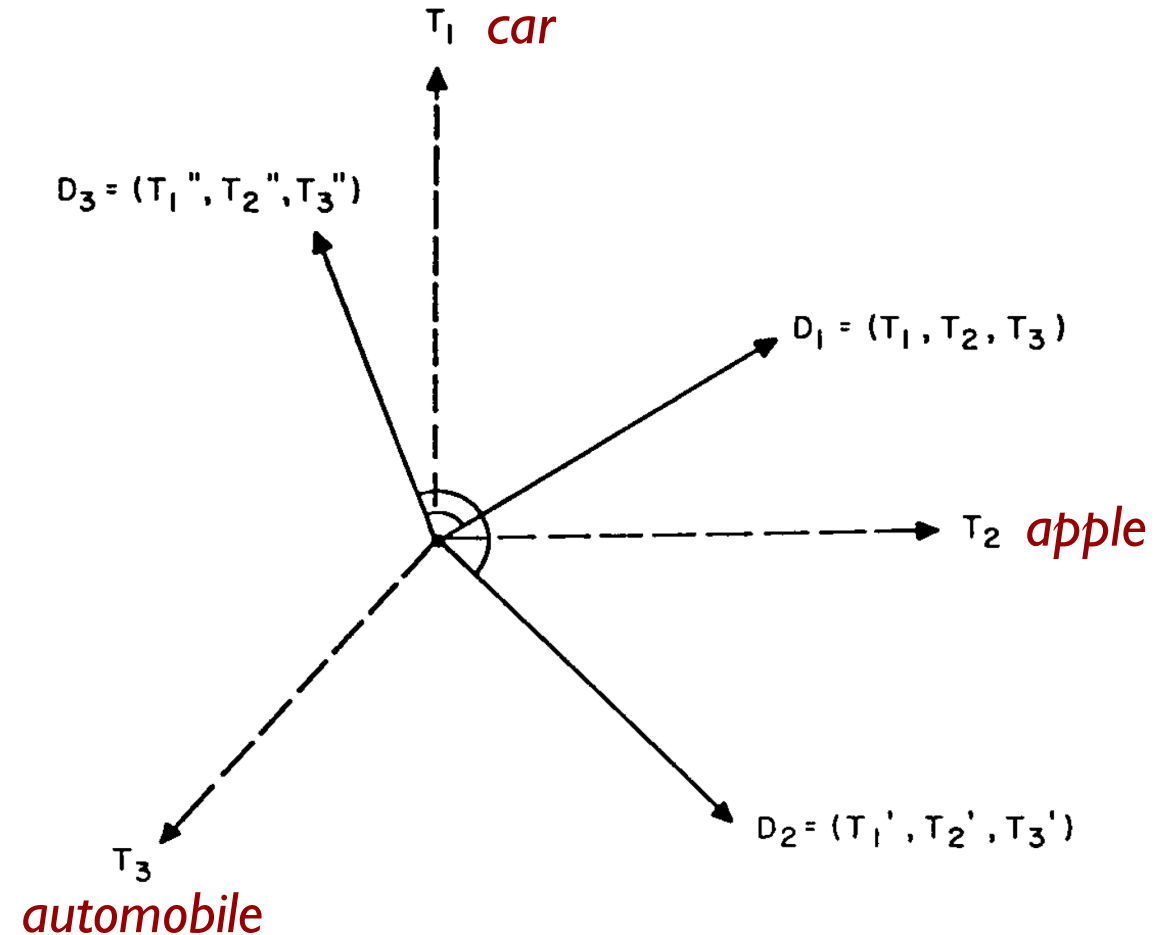
	<i>large</i>	<i>language</i>	<i>models</i>	<i>chatgpt</i>	<i>generative</i>	<i>openai</i>
<i>Doc 1</i>	1	1	1	0	0	0
<i>Doc 2</i>	0	0	0	1	1	1

- Doc 1** and **Doc 2** are totally irrelevant according to TF-IDF and BM25. Why?
  - Because we directly use the corresponding row to represent each document.
  - Doc 1**: [1, 1, 1, 0, 0, 0]
  - Doc 2**: [0, 0, 0, 1, 1, 1]
  - The inner product or cosine similarity between them is 0.

# Recap: Vector Space Model

- A document is more relevant to a query if they are more “similar” in the vector space.
- What are the axes?
- How about each dimension representing **a word in the vocabulary**?
  - In this case, we assume that **any two distinct words are orthogonal to each other!**
  - $\cos(car, apple) = 0$
  - $\cos(car, automobile) = 0$
  - $\cos(car, car) = 1$

Fig. 1. Vector representation of document space.

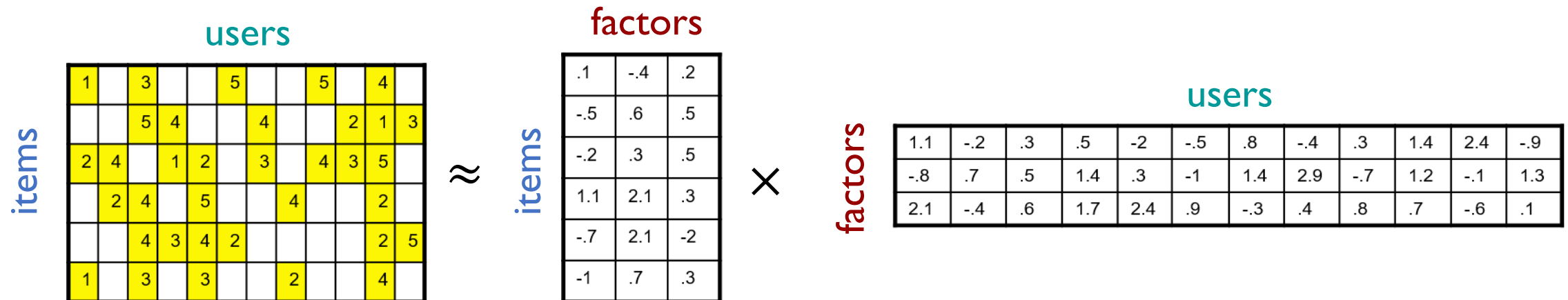


# How about Recommender Systems?

- If we were to adopt the idea of lexical matching, then

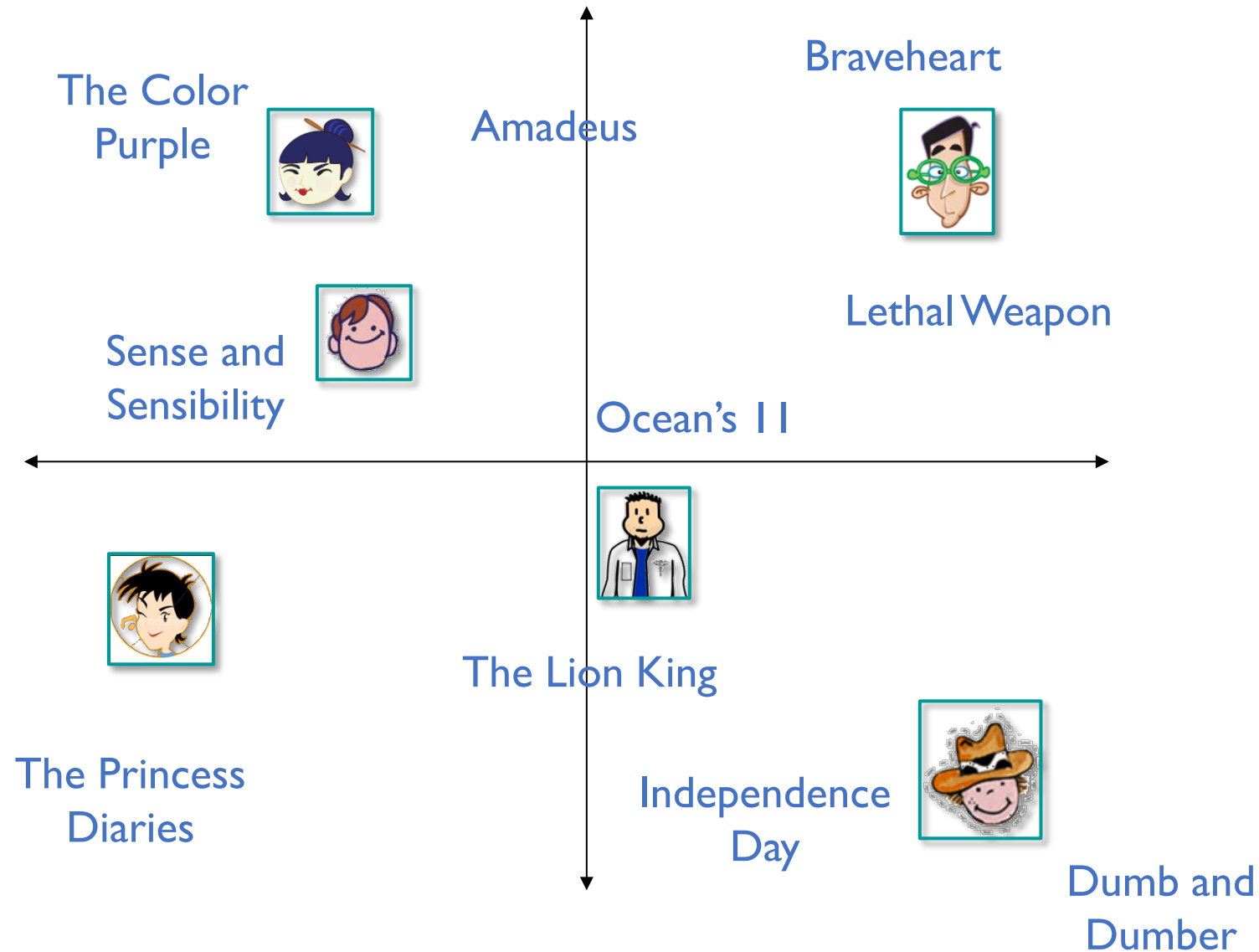
$$U_{xi} = \begin{cases} 1, & \text{if user } x \text{ has purchased item } i \\ 0, & \text{otherwise} \end{cases}$$

- The task of a recommender system is to suggest **items that the user has not purchased**. If we were to predict all unpurchased items as 0, **there would be nothing to recommend**. Therefore, the methods we introduced earlier clearly do not operate in this manner.
- What did we do?



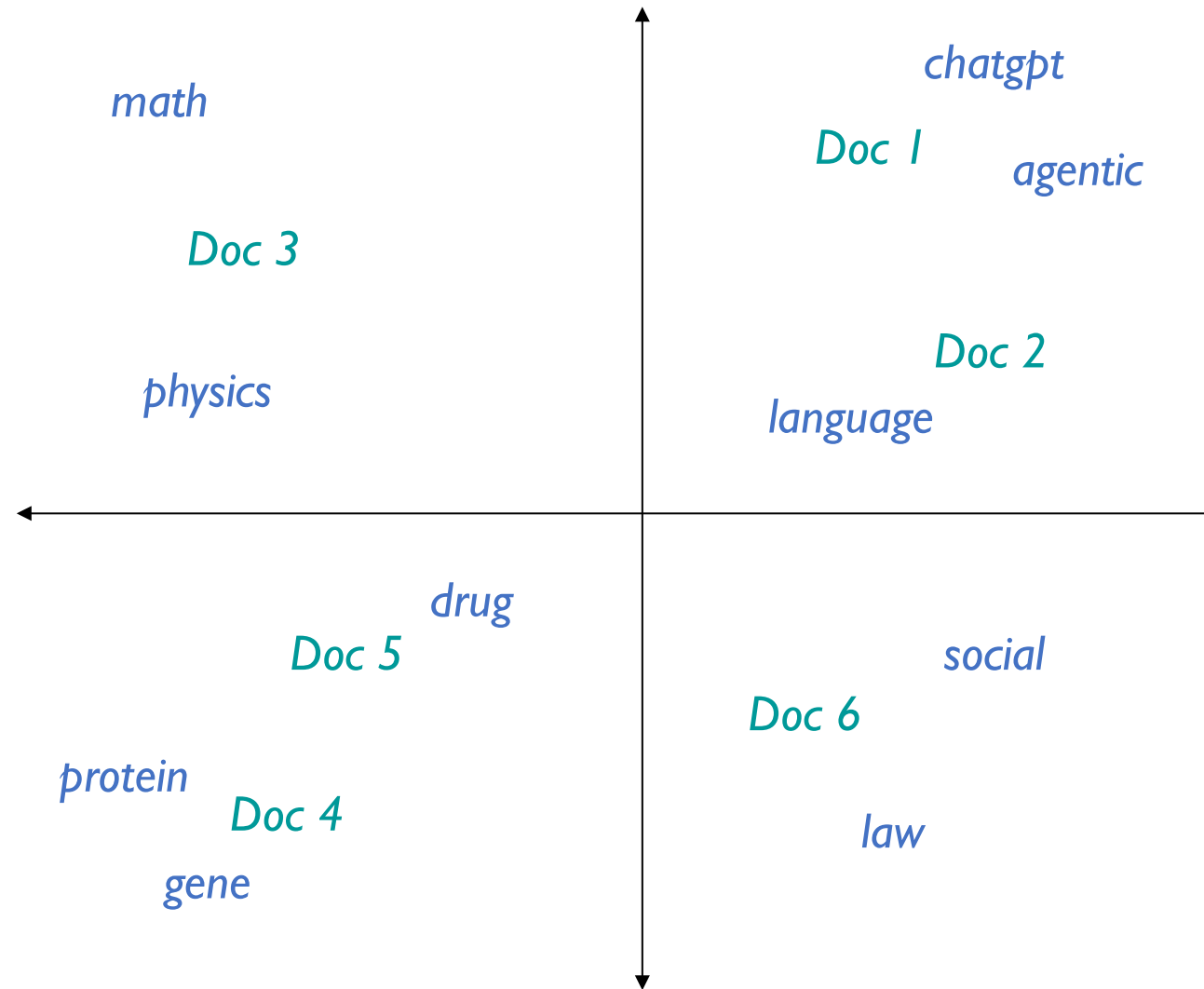


# Latent Space for Users and Items

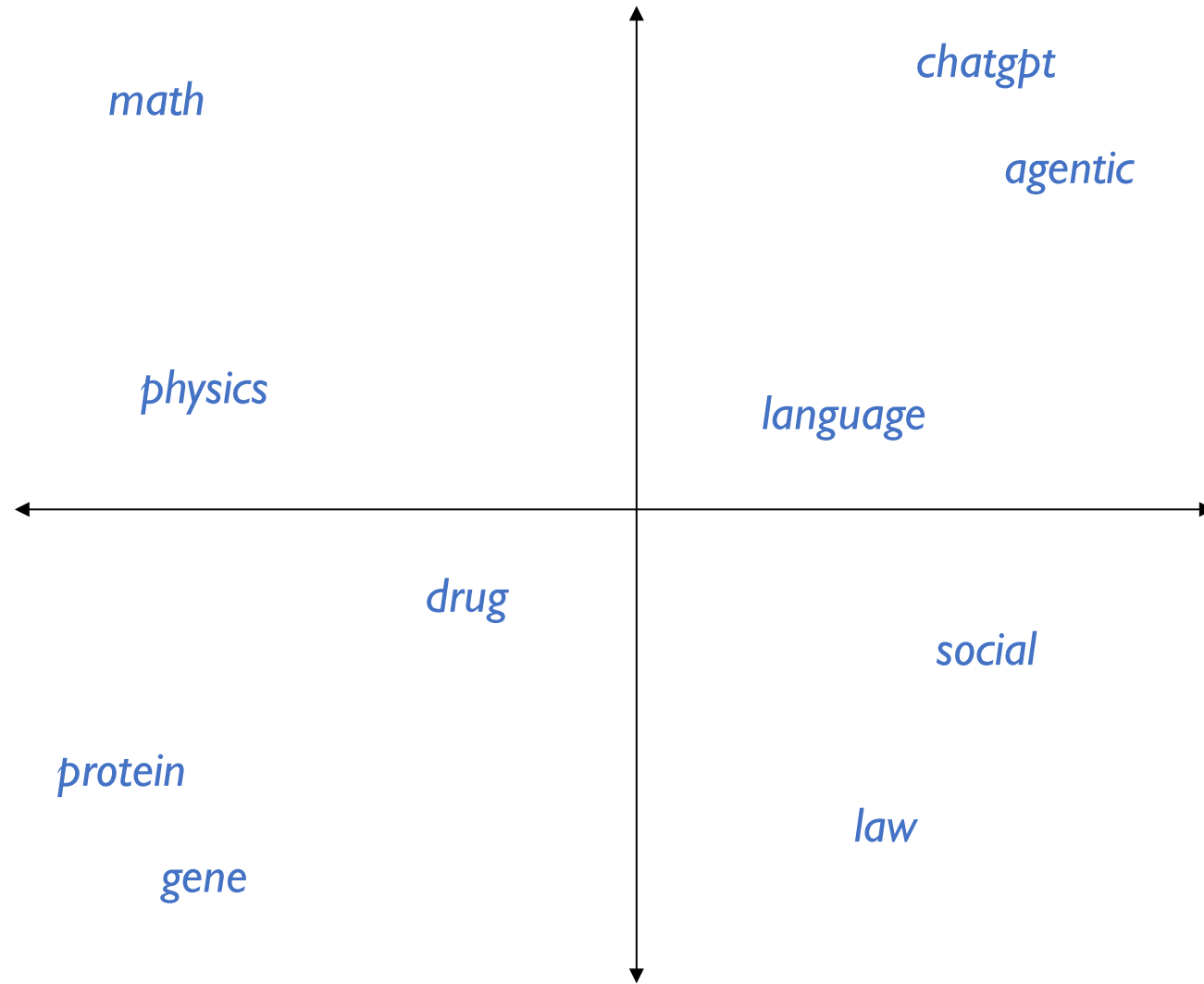


# Latent Space for Words and Documents?

- Factorizing the Document-Word Occurrence Matrix
- PTE [Tang et al., KDD 2015]



# Latent Space for Words Only?

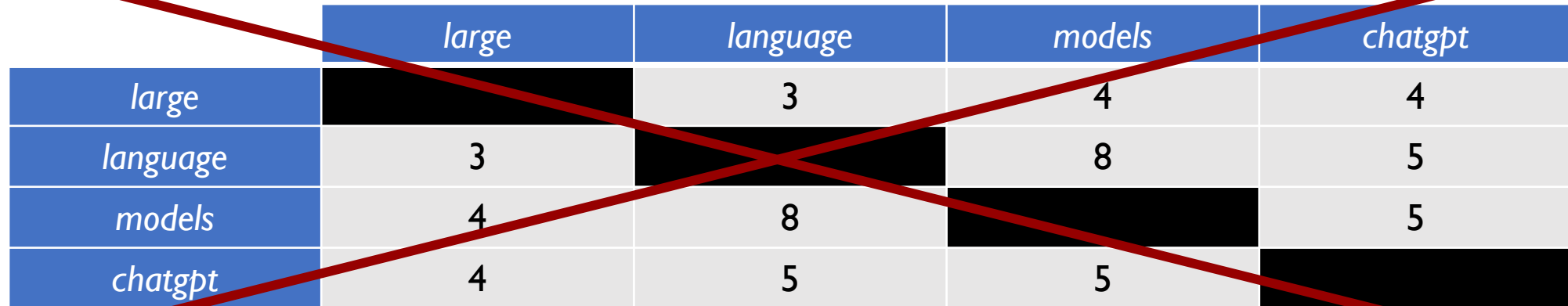


# Latent Space for Words Only

- **Question 1:** Given a new document or a new query, how can we derive its vector?
  - Average of word vectors
  - $\text{Vector}(\text{"large language models"}) = \frac{1}{3} (\text{Vector}(\text{large}) + \text{Vector}(\text{language}) + \text{Vector}(\text{models}) )$
- **Question 2:** Which matrix shall we factorize in this case?
  - Word-word co-occurrence matrix
  - How do we define “co-occur”?
    - Two words “co-occur” if they are in the same document.
    - Two words “co-occur” if they are no more 5 words apart in a document.
      - word2vec [Mikolov et al., NIPS 2013]
      - GloVe [Pennington et al., EMNLP 2014]
  - ...

# Word-Word Co-occurrence Matrix

- In your imagination, this matrix might look as follows:



	<i>large</i>	<i>language</i>	<i>models</i>	<i>chatgpt</i>
<i>large</i>		3	4	4
<i>language</i>	3		8	5
<i>models</i>	4	8		5
<i>chatgpt</i>	4	5	5	

- [Levy and Golberg, NIPS 2014] word2vec is **mathematically equivalent** to factorizing a word-word matrix  $U$ , where

$$U_{xy} = \log \frac{\#(x, y) |\mathcal{D}|}{\#(x) \#(y)} - \log b$$

# Next Lecture: Word Embedding, particularly word2vec

## Announcing the NeurIPS 2023 Paper Awards

[COMMUNICATIONS CHAIRS 2023](#) / [2023 Conference](#) / [awards, neurips2023](#)

### Test of Time

This year, following the usual practice, we chose a NeurIPS paper from 10 years ago to receive the Test of Time Award, and **“Distributed Representations of Words and Phrases and their Compositionality”** by Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, won.

Published at NeurIPS 2013 and cited over 40,000 times, the work introduced the seminal word embedding technique word2vec. Demonstrating the power of learning from large amounts of unstructured text, the work catalyzed progress that marked the beginning of a new era in natural language processing.

# Next Lecture: Word Embedding, particularly word2vec

- “... Demonstrating *the power of learning from large amounts of unstructured text*, the work catalyzed progress that *marked the beginning of a new era in natural language processing*.”
- What should you do when you have no labels at all but still want to find a classification task to train your model?
  - Mask a word in your text and predict it using its context.

*ChatGPT is a [MASK] artificial intelligence chatbot*



*generative*

- What is the label space in this classification task?
  - The entire vocabulary!



Thank You!

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F25.html>