



CSCE 670 - Information Storage and Retrieval

Week 10: BERT-Based Ranking (Cont'd), Neural Recommendation

Yu Zhang

yuzhang@tamu.edu

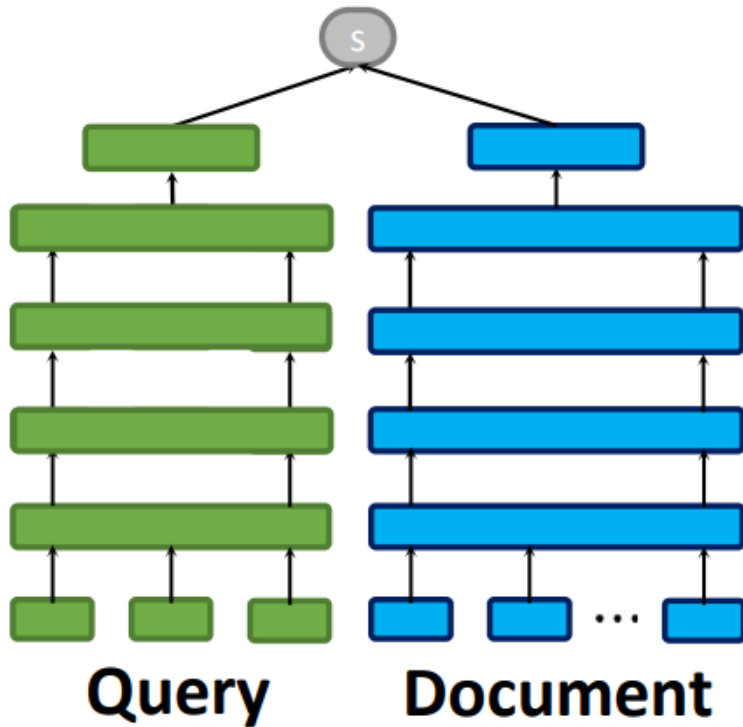
Course Website: <https://yuzhang-teaching.github.io/CSCE670-F26.html>

Quiz 3!

- All policies are the same as Quiz 1 (number of questions, time limit, grading, etc.)
- Scope:
 - Week 7 (Bayesian Personalized Ranking), **starting from Slide 27 (titled “Our Task”)**
 - Week 8 (word2vec, Neural Ranking)
 - Week 9 (BERT, BERT-Based Ranking)
 - Homework 2
 - **None of the content in Week 10 slides (e.g., ColBERT) will be covered.**

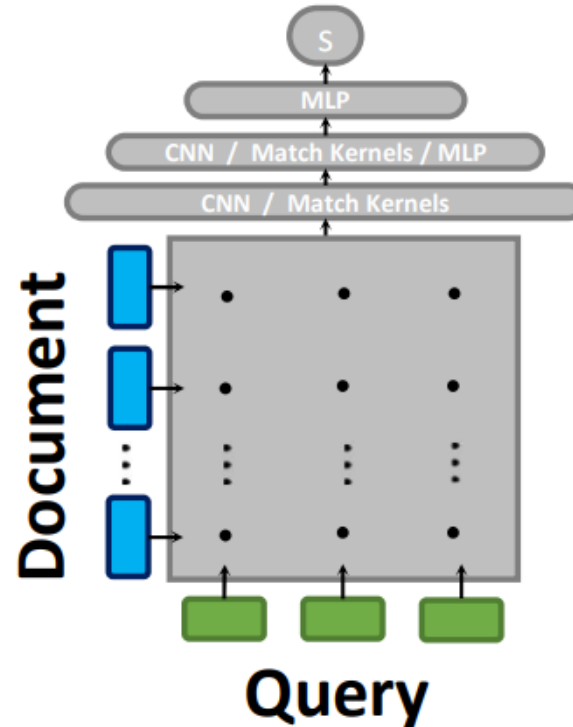
BERT-Based Ranking: ColBERT

Previously Introduced Neural Ranking Paradigms



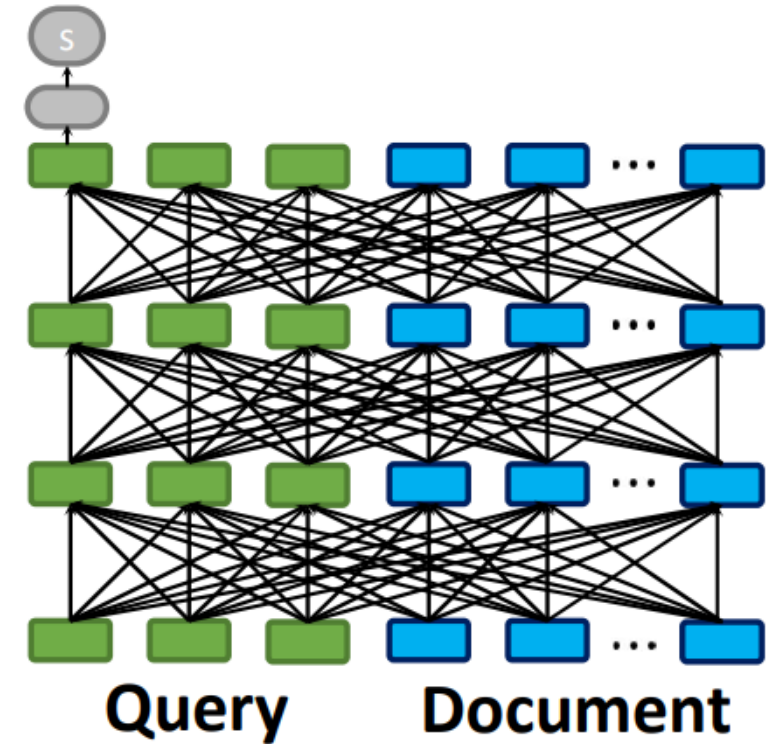
(a) Representation-based Similarity

- DESM, DPR, BERT Bi-Encoder
- Efficient
- Independent Query/Doc encoding



(b) Query-Document Interaction

- Duet, Conv-KNRM
- Effective but expensive
- $\text{context}(\text{Query}) = \text{Doc}$

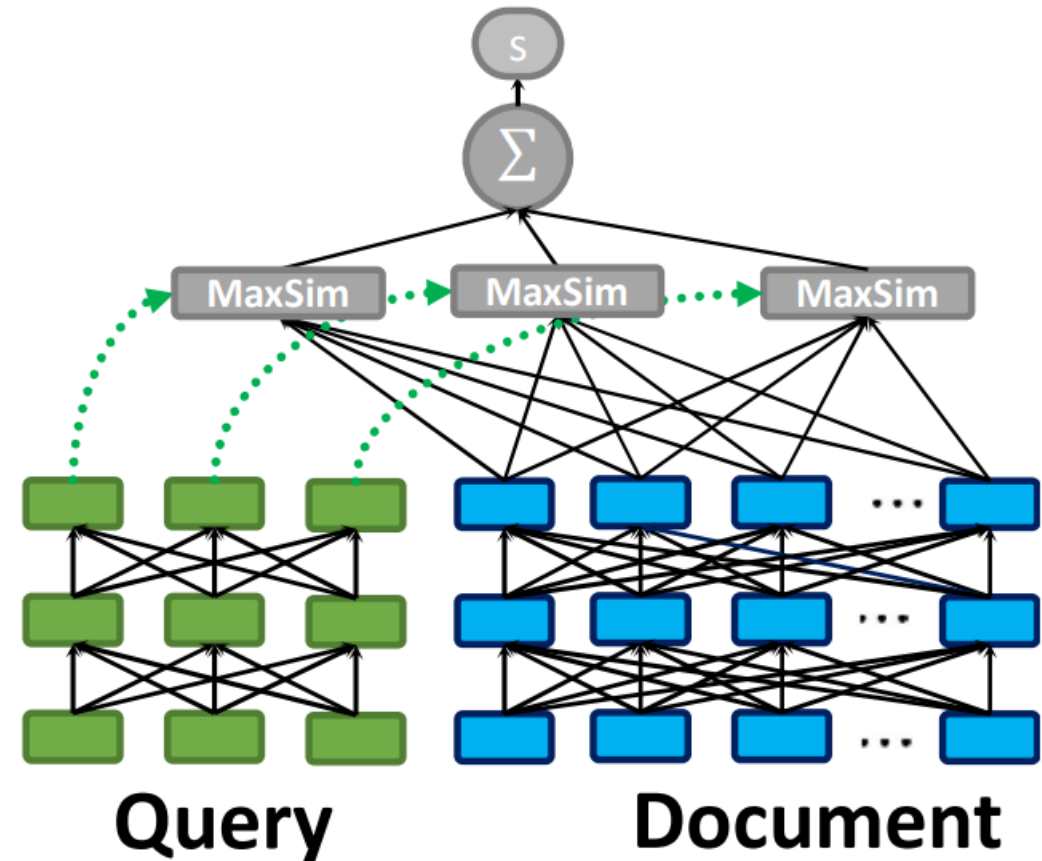


(c) All-to-all Interaction

- BERT Cross-Encoder
- Effective but expensive
- $\text{context}(\text{Query}) = \text{Query} \ \& \ \text{Doc}$

Late Interaction

- Can we keep precomputation and still have **fine-grained query-document** interactions?
- Desired Properties:
 - Independent encoding
 - Fine-grained representations
 - End-to-end retrieval (the retrieval-reranking paradigm can be used, but it is not mandatory)



(d) Late Interaction

ColBERT [Khattab and Zaharia, SIGIR 2020]

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

Omar Khattab
Stanford University
okhattab@stanford.edu

Matei Zaharia
Stanford University
matei@cs.stanford.edu

ABSTRACT

Recent progress in Natural Language Understanding (NLU) is driving fast-paced advances in Information Retrieval (IR), largely owed to fine-tuning deep language models (LMs) for document ranking. While remarkably effective, the ranking models based on these LMs increase computational cost by orders of magnitude over prior approaches, particularly as they must feed each query–document pair through a massive neural network to compute a single relevance score. To tackle this, we present ColBERT, a novel ranking model that adapts deep LMs (in particular, BERT) for efficient retrieval. ColBERT introduces a *late interaction* architecture that independently encodes the query and the document using BERT and then employs a cheap yet powerful interaction step that models their

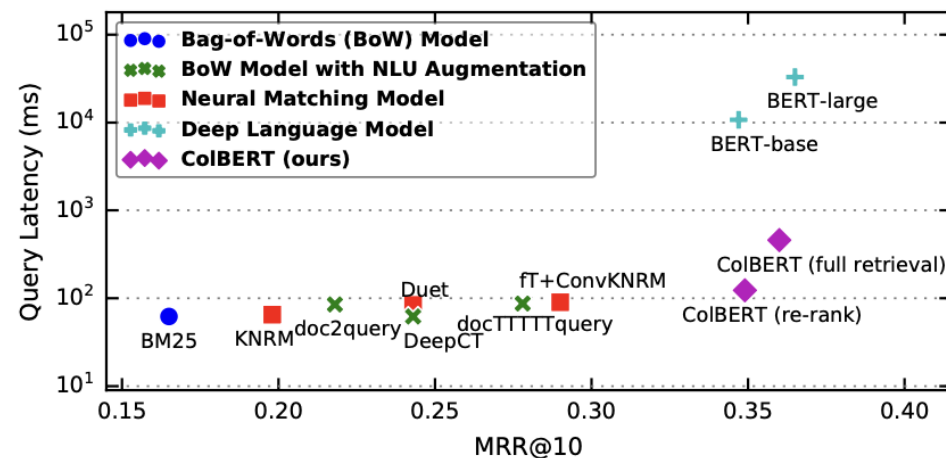


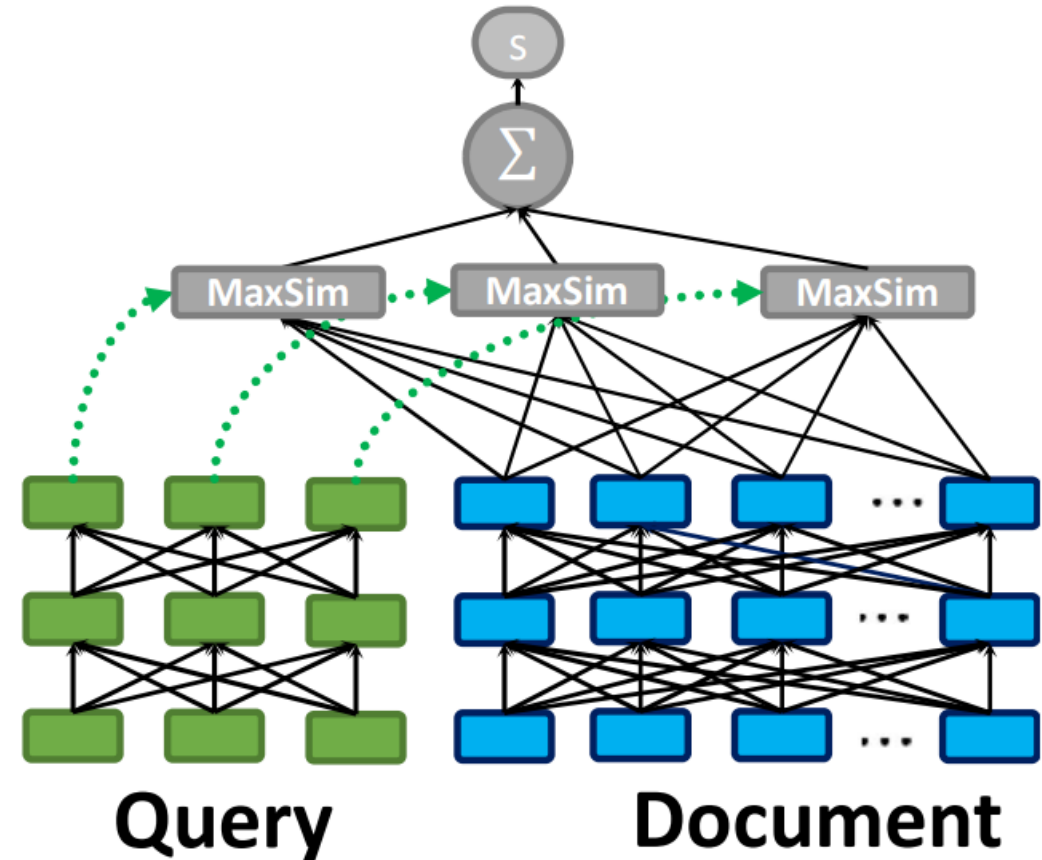
Figure 1: Effectiveness (MRR@10) versus Mean Query Latency (log-scale) for a number of representative ranking models on MS MARCO Ranking [41]. The figure also shows

Late Interaction

- **Step 1:** Encode the **document** into a sequence of vectors $\mathbf{e}_{d_1}, \mathbf{e}_{d_2}, \dots, \mathbf{e}_{d_L}$
- **Step 2:** Encode the **query** into a sequence of vectors $\mathbf{e}_{q_1}, \mathbf{e}_{q_2}, \dots, \mathbf{e}_{q_M}$
- **Step 3:**

$$\text{score}(q, d) = \sum_{m=1}^M \max_{1 \leq l \leq L} \mathbf{e}_{q_m}^T \mathbf{e}_{d_l}$$

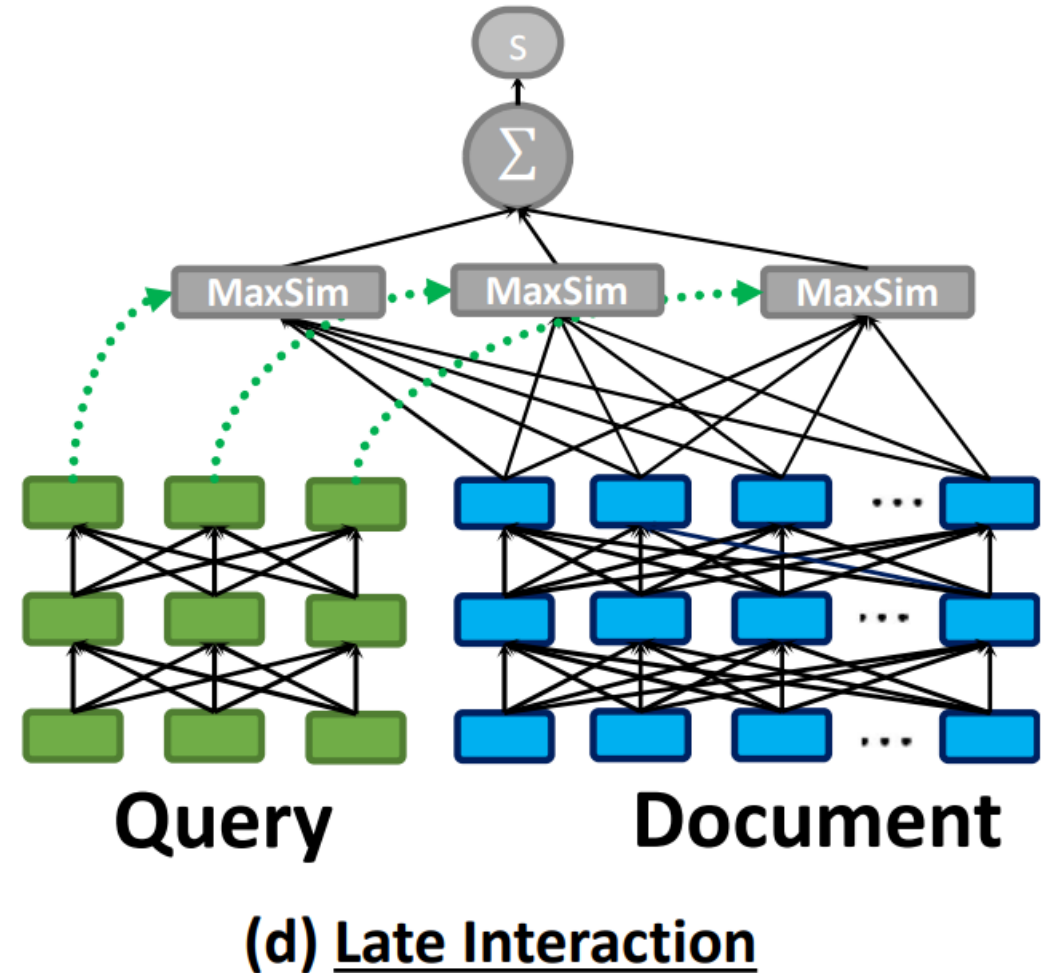
- **Intuition:** For each **word** in the **query**, find the most matching **word** in the current candidate **document** to compute the similarity



(d) Late Interaction

Late Interaction

- Example
 - **Query** representation
 $[1, 0], [0, 1]$
 - **Document** representation
 $[1, 1], [0.5, 0.5]$
 - For $[1, 0]$, which **document** vector is the most similar?
 - For $[0, 1]$, which **document** vector is the most similar?
 - $\text{score}(q, d)$
 $= [1, 0] \begin{bmatrix} 1 \\ 1 \end{bmatrix} + [0, 1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2$



Late Interaction: Real Example of Matching

when did the transformers cartoon series come out?

[...] the animated [...] The Transformers [...] [...] It was released [...] **on** August 8, 1986

when did the transformers cartoon series come out?

[...] the animated [...] The **Transformers** [...] [...] It was released [...] on August 8, 1986

when did the transformers cartoon series come out?

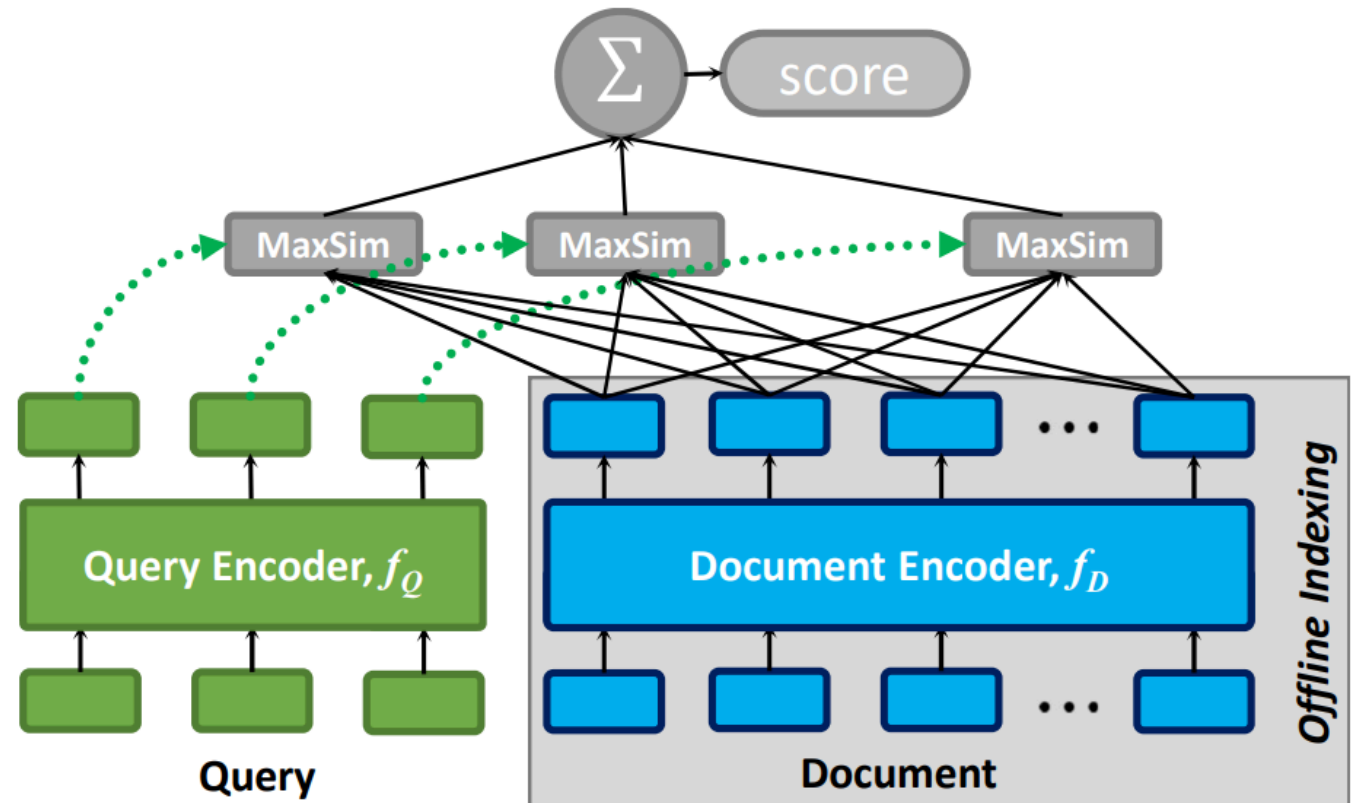
[...] the **animated** [...] The Transformers [...] [...] It was released [...] on August 8, 1986

when did the transformers cartoon series come out?

[...] the animated [...] The Transformers [...] [...] It was **released** [...] on August 8, 1986

Offline Indexing

- For Bi-Encoder, index **one vector** for each candidate document
- For ColBERT, index **a sequence of vectors** for each candidate document



Performance of ColBERT: Reranking

- On par with BERT-Base (Cross-Encoder)
- Slightly worse than BERT-Large (Cross-Encoder)
- BUT much faster!

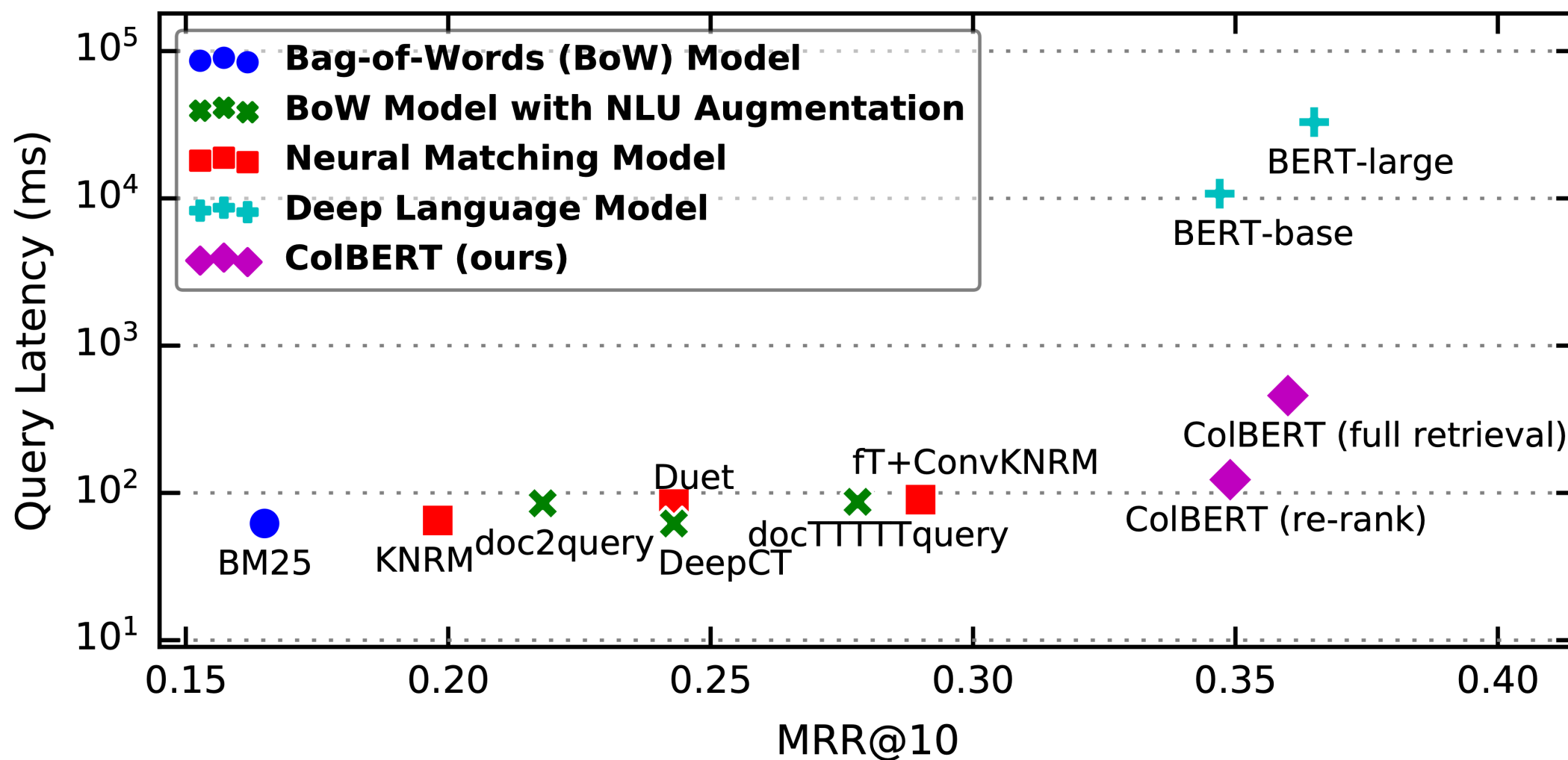
Method	MRR@10 (Dev)	MRR@10 (Eval)	Re-ranking Latency (ms)	FLOPs/query
BM25 (official)	16.7	16.5	-	-
KNRM	19.8	19.8	3	592M (0.085×)
Duet	24.3	24.5	22	159B (23×)
fastText+ConvKNRM	29.0	27.7	28	78B (11×)
BERT _{base} [25]	34.7	-	10,700	97T (13,900×)
BERT _{base} (our training)	36.0	-	10,700	97T (13,900×)
BERT _{large} [25]	36.5	35.9	32,900	340T (48,600×)
ColBERT (over BERT _{base})	34.9	34.9	61	7B (1×)

Performance of ColBERT: End-to-End Retrieval

- $<10\times$ slower than BM25 and Bi-Encoder ranking models
- BUT much more effective!

Method	MRR@10 (Dev)	MRR@10 (Local Eval)	Latency (ms)	Recall@50	Recall@200	Recall@1000
BM25 (official)	16.7	-	-	-	-	81.4
BM25 (Anserini)	18.7	19.5	62	59.2	73.8	85.7
doc2query	21.5	22.8	85	64.4	77.9	89.1
DeepCT	24.3	-	62 (<i>est.</i>)	69 [2]	82 [2]	91 [2]
docTTTTTquery	27.7	28.4	87	75.6	86.9	94.7
ColBERT _{L2} (re-rank)	34.8	36.4	-	75.3	80.5	81.4
ColBERT _{L2} (end-to-end)	36.0	36.7	458	82.9	92.3	96.8

Striking a Good Balance Between Bi-Encoder and Cross-Encoder



Robustness: Out-of-Domain Quality

- So far, we have looked at **in-domain** effectiveness evaluations
 - We had training and evaluation data for MS MARCO
 - We often want to use retrieval in new, **out-of-domain** settings with NO training data and NO validation data
 - This is sometimes called a “**zero-shot**” setting; it emphasizes **generalization**
 - **BEIR** is a popular benchmark for IR models in “zero-shot” scenarios



BEIR

Split (→)					Train	Dev	Test			Avg. Word Lengths	
Task (↓)	Domain (↓)	Dataset (↓)	Title	Relevancy	#Pairs	#Query	#Query	#Corpus	Avg. D / Q	Query	Document
Passage-Retrieval	Misc.	MS MARCO [45]	✗	Binary	532,761	—	6,980	8,841,823	1.1	5.96	55.98
Bio-Medical Information Retrieval (IR)	Bio-Medical	TREC-COVID [65]	✓	3-level	—	—	50	171,332	493.5	10.60	160.77
	Bio-Medical	NFCorpus [7]	✓	3-level	110,575	324	323	3,633	38.2	3.30	232.26
	Bio-Medical	BioASQ [61]	✓	Binary	32,916	—	500	14,914,602	4.7	8.05	202.61
Question Answering (QA)	Wikipedia	NQ [34]	✓	Binary	132,803	—	3,452	2,681,468	1.2	9.16	78.88
	Wikipedia	HotpotQA [76]	✓	Binary	170,000	5,447	7,405	5,233,329	2.0	17.61	46.30
	Finance	FiQA-2018 [44]	✗	Binary	14,166	500	648	57,638	2.6	10.77	132.32
Tweet-Retrieval	Twitter	Signal-1M (RT) [59]	✗	3-level	—	—	97	2,866,316	19.6	9.30	13.93
News Retrieval	News	TREC-NEWS [58]	✓	5-level	—	—	57	594,977	19.6	11.14	634.79
	News	Robust04 [64]	✗	3-level	—	—	249	528,155	69.9	15.27	466.40
Argument Retrieval	Misc.	ArguAna [67]	✓	Binary	—	—	1,406	8,674	1.0	192.98	166.80
	Misc.	Touché-2020 [6]	✓	3-level	—	—	49	382,545	19.0	6.55	292.37
Duplicate-Question Retrieval	StackEx.	CQADupStack [25]	✓	Binary	—	—	13,145	457,199	1.4	8.59	129.09
	Quora	Quora	✗	Binary	—	5,000	10,000	522,931	1.6	9.53	11.44
Entity-Retrieval	Wikipedia	DBpedia [21]	✓	3-level	—	67	400	4,635,922	38.2	5.39	49.68
Citation-Prediction	Scientific	SCIDOCS [9]	✓	Binary	—	—	1,000	25,657	4.9	9.38	176.19
Fact Checking	Wikipedia	FEVER [60]	✓	Binary	140,085	6,666	6,666	5,416,568	1.2	8.13	84.76
	Wikipedia	Climate-FEVER [14]	✓	Binary	—	—	1,535	5,416,593	3.0	20.13	84.76
	Scientific	SciFact [68]	✓	Binary	920	—	300	5,183	1.1	12.37	213.63

Robustness: Out-of-Domain NDCG@10

IR Task	Classical IR BM25	Interaction Models ELECTRA re-ranker	Representation Similarity DPR	Representation Similarity SBERT	Late Interaction CoBERT
BioMed	48	49	22	34	49
QA	38	51	33	41	48
Tweet	39	31	16	26	27
News	37	43	16	37	39
Arguments	52	35	15	34	25
Duplicates	53	56	20	58	60
Entity	29	38	26	34	39
Citation	16	15	8	13	15
Fact-Check	48	52	34	47	54
Overall Avg	42	45	23	39	44

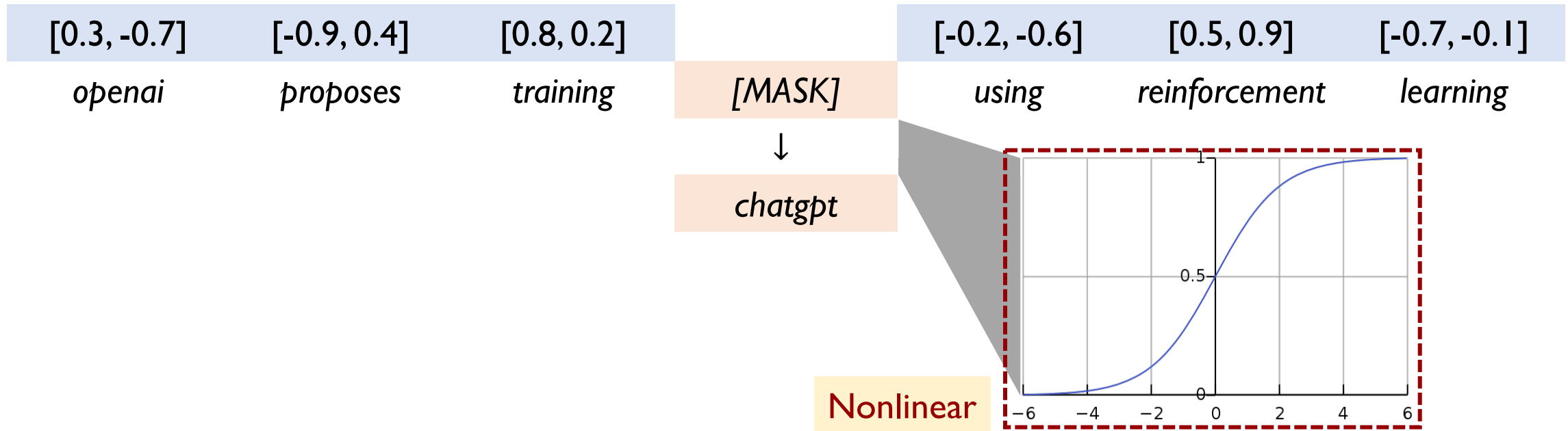
Robustness: Out-of-Domain Recall@100

IR Task	Classical IR BM25	Interaction Models ELECTRA re-ranker	Representation Similarity DPR	Representation Similarity SBERT	Late Interaction CoBERT
BioMed	45	45	23	35	45
QA	67	67	60	68	75
Tweet	38	38	16	26	28
News	40	40	22	37	37
Arguments	70	70	46	62	61
Duplicates	77	77	44	79	81
Entity	38	38	35	40	46
Citation	35	35	22	30	34
Fact-Check	71	71	65	74	75
Overall Avg	59	59	43	57	61

How to build effective “zero-shot” rankers?
LLMs! (Next week)

Neural Recommendation: Neural Collaborative Filtering

Neural Word Embeddings (word2vec)



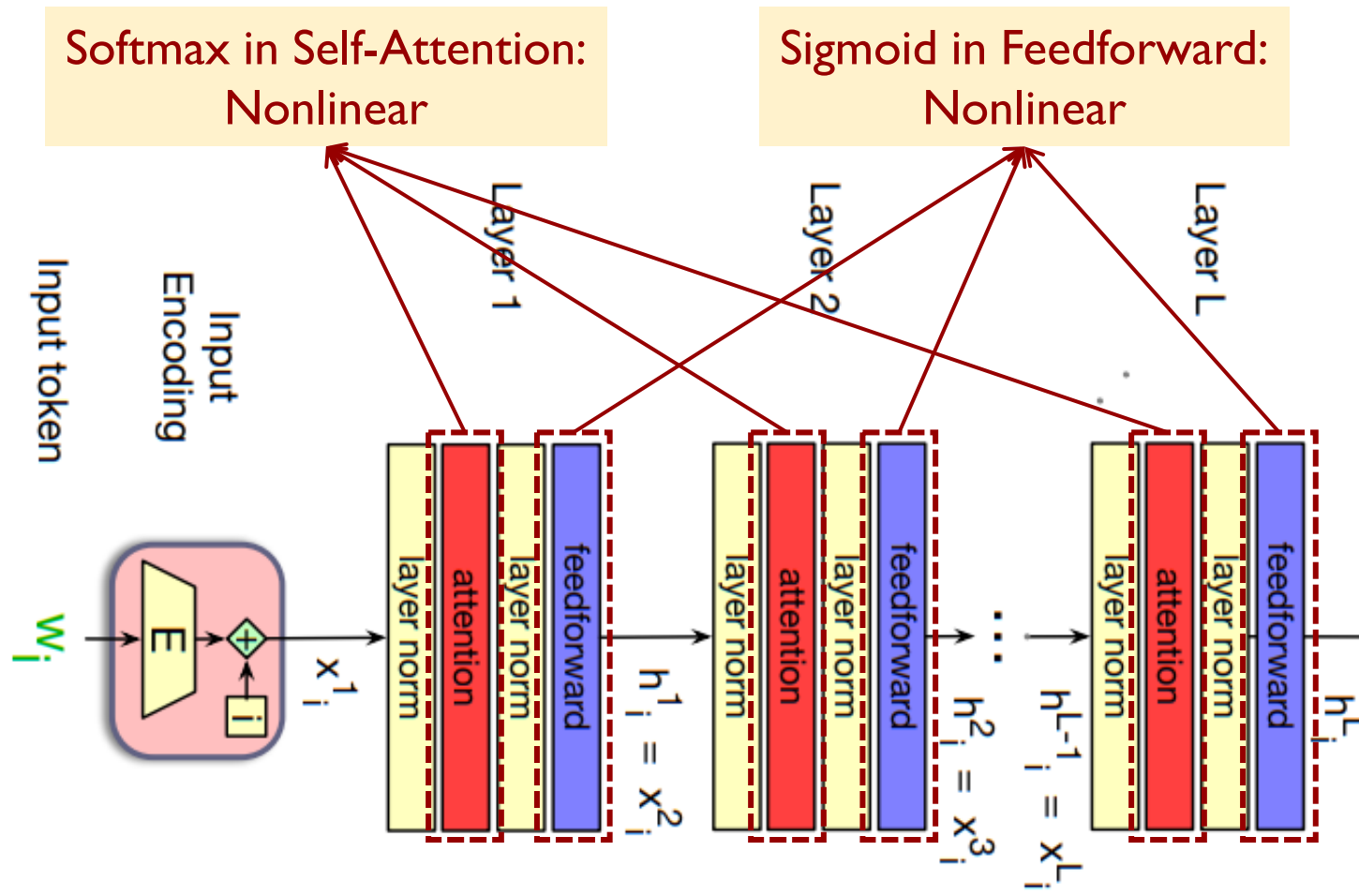
- [Levy and Golberg, NIPS 2014] word2vec is mathematically equivalent to factorizing a word-word matrix U , where

Nonlinear

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

Neural Language Models (BERT)

- Transformer is a neural network



Nonlinearity in Recommender Systems?

- Do the recommender system techniques we have learned so far contain enough **nonlinearity**?
 - No, matrix factorization is essentially modeling the inner product of vectors, which is a highly linear operation.

users

1		3		5		5		4		
		5	4			4		2	1	3
2	4		1	2		3		4	3	5
	2	4		5		4			2	
		4	3	4	2				2	5
1		3		3		2		4		

items

U

$\approx$

factors

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

items

Q

$\times$

users

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

factors

P^T

$$U_{xi} = q_i p_x^T$$

Neural Collaborative Filtering [He et al., WWW 2017]

Neural Collaborative Filtering*

Xiangnan He
National University of
Singapore, Singapore
xiangnanhe@gmail.com

Liqiang Nie
Shandong University
China
nieliqiang@gmail.com

Lizi Liao
National University of
Singapore, Singapore
liaolizi.llz@gmail.com

Xia Hu
Texas A&M University
USA
hu@cse.tamu.edu

Hanwang Zhang
Columbia University
USA
hanwangzhang@gmail.com

Tat-Seng Chua
National University of
Singapore, Singapore
dcscts@nus.edu.sg

ABSTRACT

In recent years, deep neural networks have yielded immense success on speech recognition, computer vision and natural language processing. However, the exploration of deep neural networks on recommender systems has received relatively less scrutiny. In this work, we strive to develop techniques based on neural networks to tackle the key problem in recommendation — collaborative filtering — on the basis of implicit feedback.

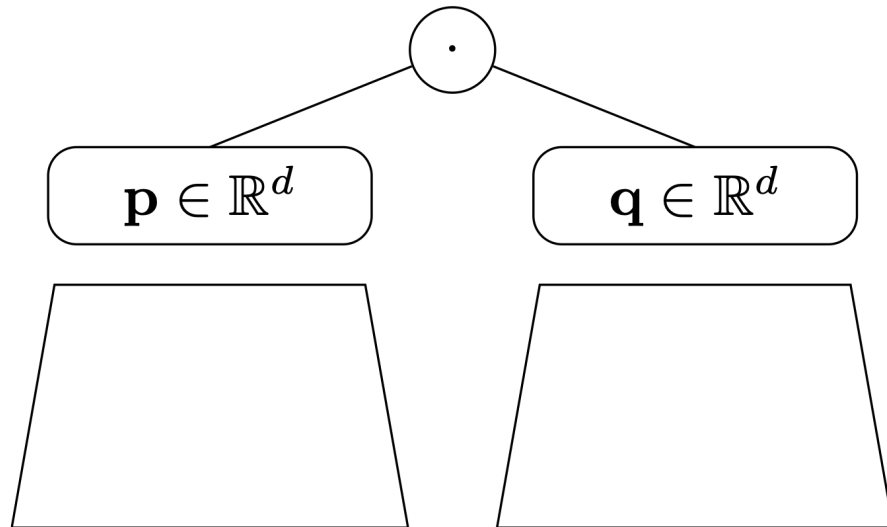
1. INTRODUCTION

In the era of information explosion, recommender systems play a pivotal role in alleviating information overload, having been widely adopted by many online services, including E-commerce, online news and social media sites. The key to a personalized recommender system is in modelling users' preference on items based on their past interactions (*e.g.*, ratings and clicks), known as collaborative filtering [31, 46]. Among the various collaborative filtering techniques, matrix

Learning the Similarity Function

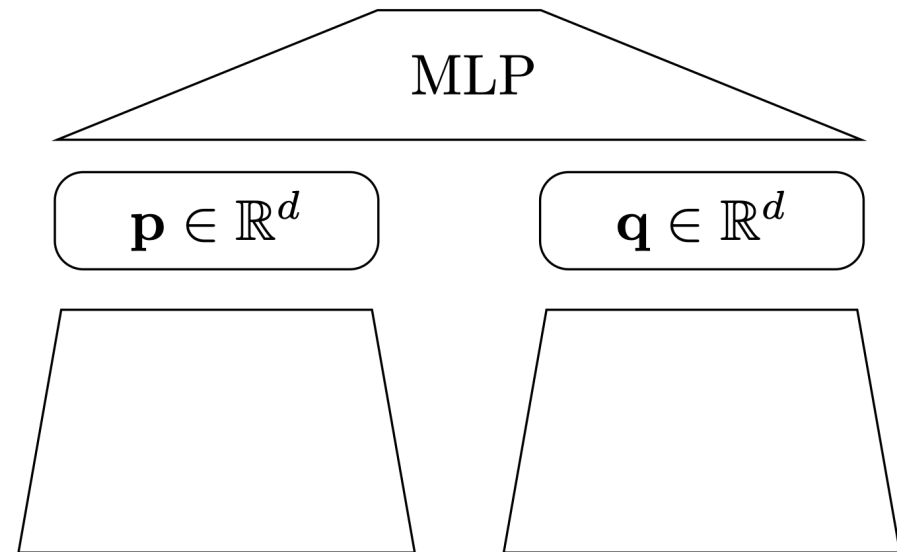
- **Predefined** similarity: inner product

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle$$



- **Learned** similarity: multi-layer perceptron (MLP)

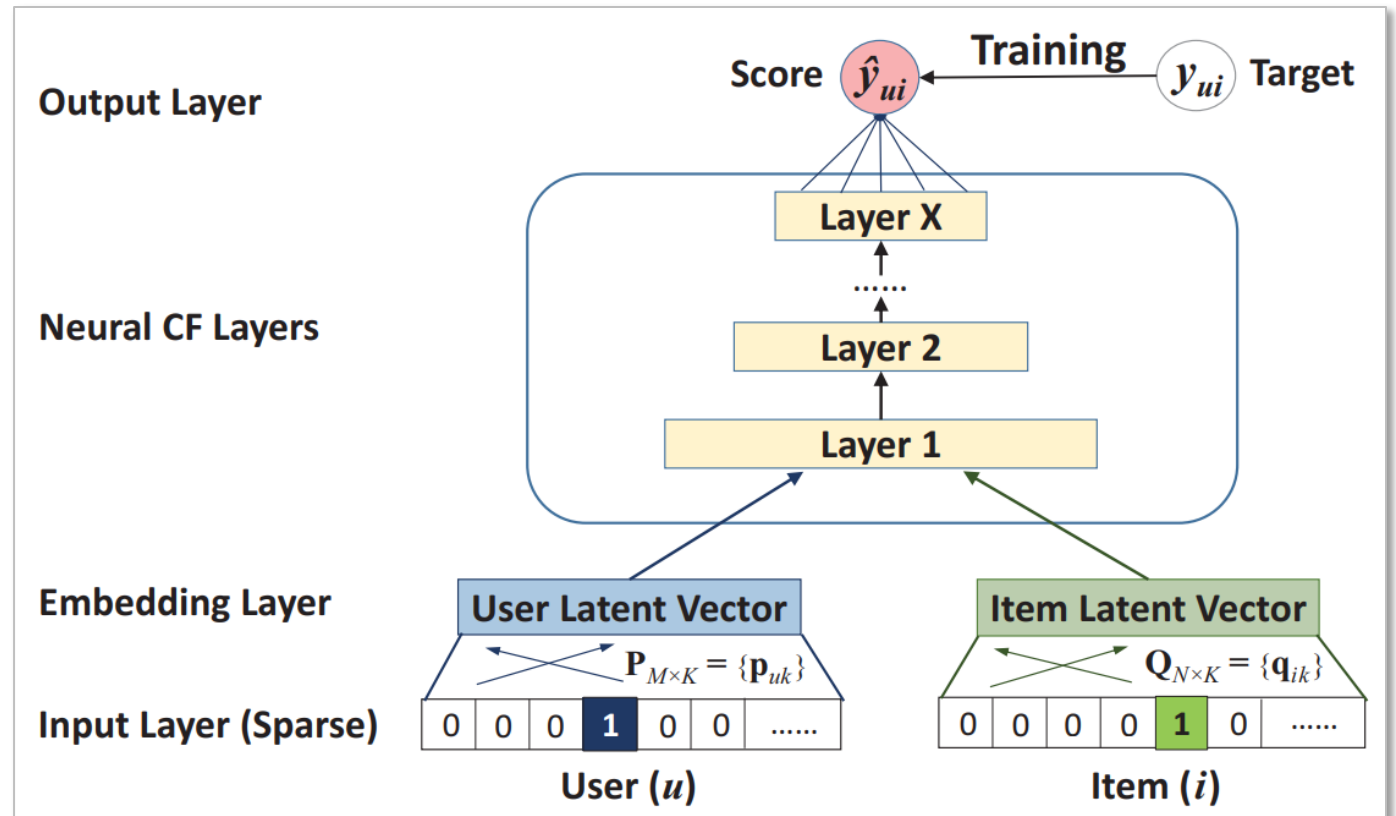
$$\phi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) = \mathbf{f}_{W_l, \mathbf{b}_l}(\dots \mathbf{f}_{W_1, \mathbf{b}_1}([\mathbf{p}, \mathbf{q}]) \dots)$$



- We could use any sort of embeddings down here (e.g., from SVD or Matrix Factorization)

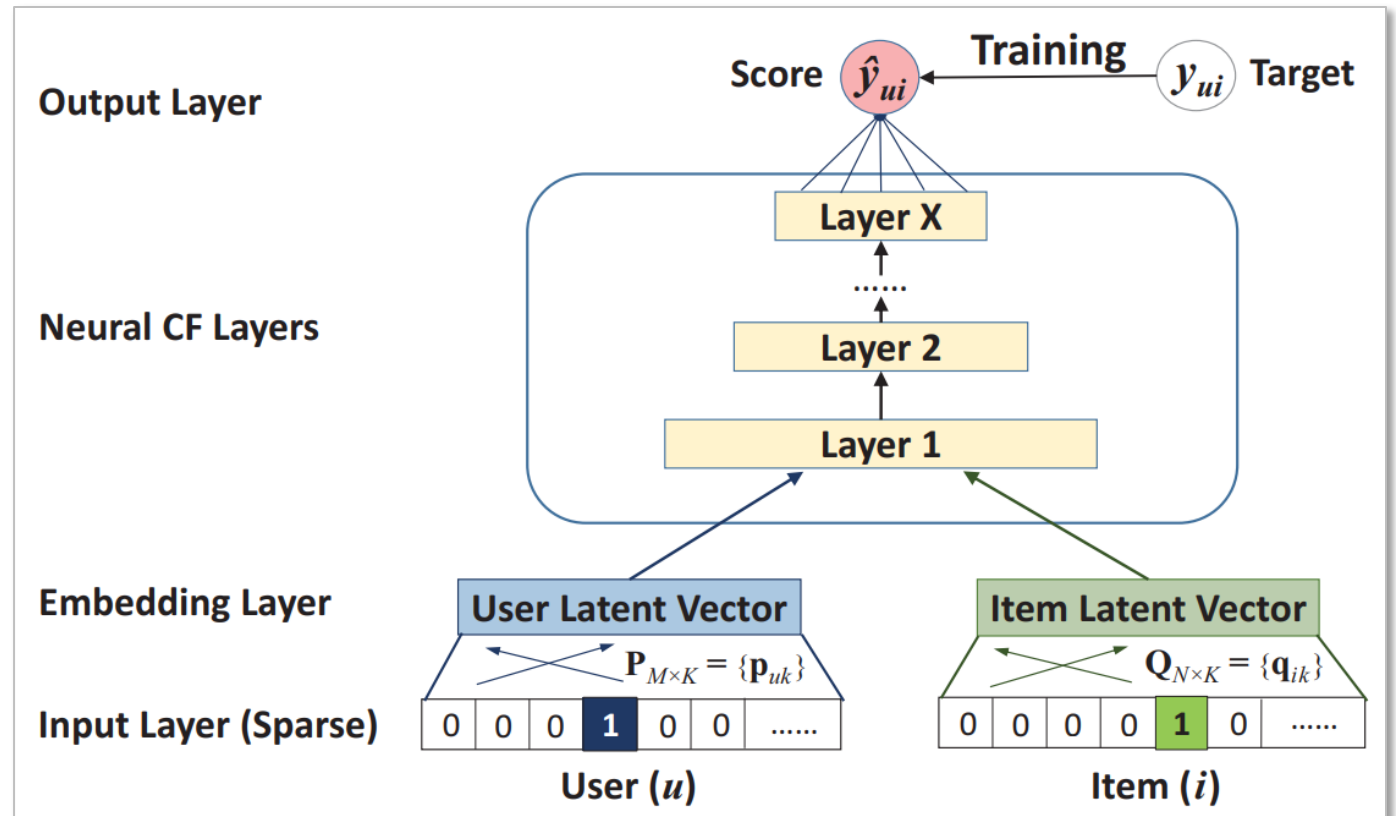
Example

- Suppose after matrix factorization,
 - User u 's vector: $\mathbf{p} = [2, 2]$
 - Item i 's vector: $\mathbf{q} = [2, 3]$
- Inner product: $\text{score}(u, i) = \mathbf{p}\mathbf{q}^T = 10$
- MLP (assuming 3 layers):
 - Input $\mathbf{x}_0 = [\mathbf{p}, \mathbf{q}] = [2, 2, 2, 3]$
 - $\mathbf{x}_1 = \text{Sigmoid}(\mathbf{W}_1\mathbf{x}_0 + \mathbf{b}_1)$
 - $\mathbf{x}_2 = \text{Sigmoid}(\mathbf{W}_2\mathbf{x}_1 + \mathbf{b}_2)$
 - $\hat{y} = \text{score}(u, i) = \text{Sigmoid}(\mathbf{W}_3\mathbf{x}_2)$



Example

- **MLP** (assuming 3 layers):
 - Input $x_0 = [p, q] = [2, 2, 2, 3]$
 - $x_1 = \text{Sigmoid}(W_1 x_0 + b_1)$
 - $x_2 = \text{Sigmoid}(W_2 x_1 + b_2)$
 - $\hat{y} = \text{Sigmoid}(W_3 x_2)$
- Parameters?
 - W_1, b_1, W_2, b_2, W_3
 - P, Q
- Learning objective?
 - $\text{RMSE} = (y - \hat{y})^2$
- Training data?
 - Known ratings in the user-item matrix



Combining Matrix Factorization and MLP

- Inner product: $\text{score}(u, i) = \mathbf{p}\mathbf{q}^T = 10$

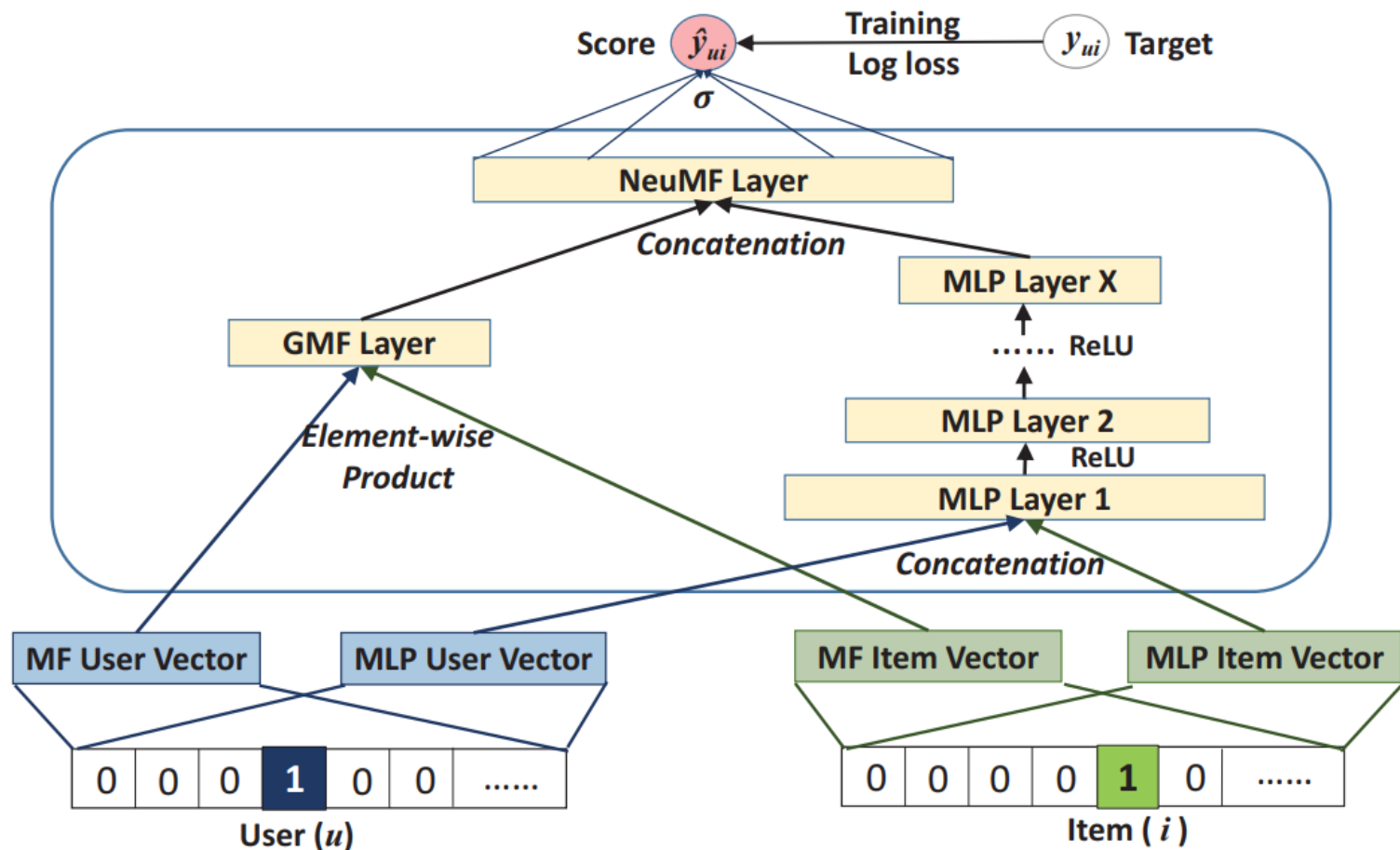
- MLP (assuming 3 layers):
 - Input $\mathbf{x}_0 = [\mathbf{p}, \mathbf{q}] = [2, 2, 2, 3]$
 - $\mathbf{x}_1 = \text{Sigmoid}(\mathbf{W}_1\mathbf{x}_0 + \mathbf{b}_1)$
 - $\mathbf{x}_2 = \text{Sigmoid}(\mathbf{W}_2\mathbf{x}_1 + \mathbf{b}_2)$
 - $\text{score}(u, i) = \text{Sigmoid}(\mathbf{W}_3\mathbf{x}_2)$

- MLP does not explicitly model the interaction between $\mathbf{p}$ and $\mathbf{q}$, but inner product does!
- Can we fuse inner product and MLP?

$$\text{score}(u, i) = \text{score}_{\text{MF}}(u, i) + \text{score}_{\text{MLP}}(u, i)$$

Combining Matrix Factorization and MLP

- User u 's vector: $\mathbf{p} = [2, 2]$
- Item i 's vector: $\mathbf{q} = [2, 3]$
- MLP: same
- Generalized Matrix Factorization (GMF):
 - Step 1: Element-wise product $\mathbf{p} \odot \mathbf{q} = [4, 6]$
 - Step 2: Nonlinear layer
 $\text{score}_{\text{MF}}(u, i) = \text{Sigmoid}(\mathbf{w} \begin{bmatrix} 4 \\ 6 \end{bmatrix})$
 - $\mathbf{w}$ are the parameters to be learned



Generalized Matrix Factorization vs. Inner Product

- User u 's vector: $\mathbf{p} = [2, 2]$
- Item i 's vector: $\mathbf{q} = [2, 3]$
- Generalized Matrix Factorization (GMF):
 - Step 1: Element-wise product $\mathbf{p} \odot \mathbf{q} = [4, 6]$
 - Step 2: Nonlinear layer $\text{score}_{\text{MF}}(u, i) = \text{Sigmoid}(\mathbf{w} \begin{bmatrix} 4 \\ 6 \end{bmatrix})$
- Suppose $\mathbf{w} = [1, -1]$, then

$$\begin{aligned}\mathbf{w}(\mathbf{p} \odot \mathbf{q}) &= [1, -1] \begin{bmatrix} 2 \times 2 \\ 2 \times 3 \end{bmatrix} = 1 \times 2 \times 2 + (-1) \times 2 \times 3 \\ &= [2, 2] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \mathbf{p} \text{diag}\{\mathbf{w}\} \mathbf{q}^T\end{aligned}$$

$$\text{score}_{\text{MF}}(u, i) = \text{Sigmoid}(\mathbf{p} \text{diag}\{\mathbf{w}\} \mathbf{q}^T)$$

Nonlinear

Learnable Parameters

Summary

Generalized Matrix Factorization:

Input $x_0 = [p, q]$

$$\begin{aligned} \text{score}_{\text{MF}}(u, i) &= \text{Sigmoid}(w(p \odot q)) \\ &= \text{Sigmoid}(p \text{ diag}\{w\} q^T) \end{aligned}$$

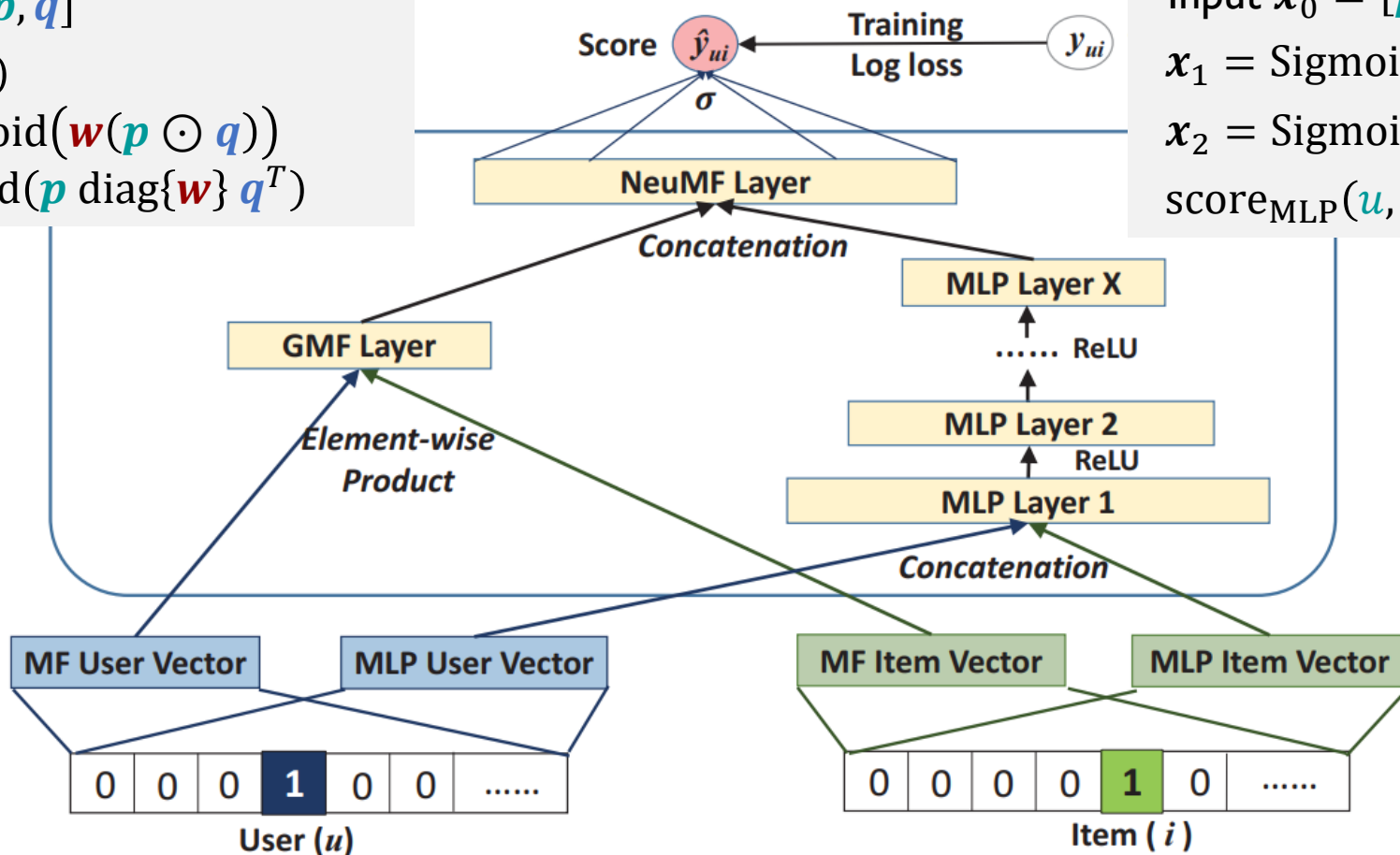
MLP (assuming 3 layers):

Input $x_0 = [p, q]$

$$x_1 = \text{Sigmoid}(W_1 x_0 + b_1)$$

$$x_2 = \text{Sigmoid}(W_2 x_1 + b_2)$$

$$\text{score}_{\text{MLP}}(u, i) = \text{Sigmoid}(W_3 x_2)$$

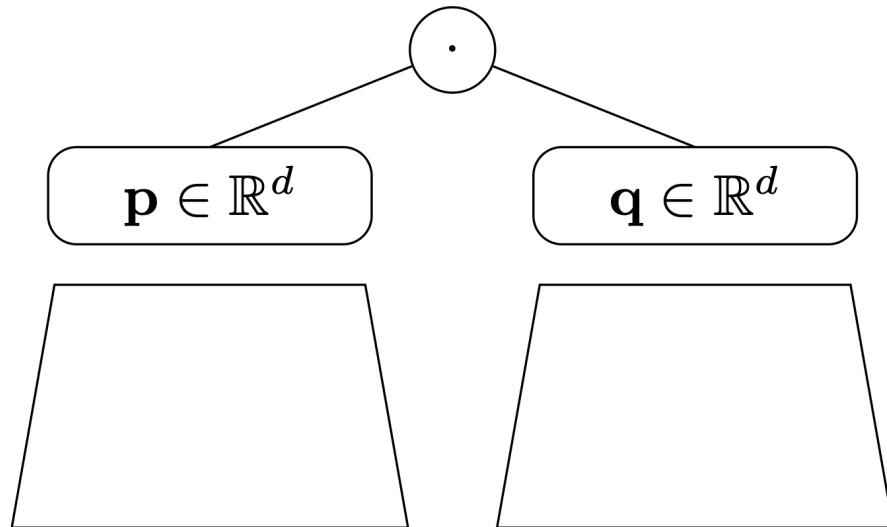


$$\text{score}(u, i) = \text{score}_{\text{MF}}(u, i) + \text{score}_{\text{MLP}}(u, i)$$

Matrix Factorization vs. Neural Collaborative Filtering

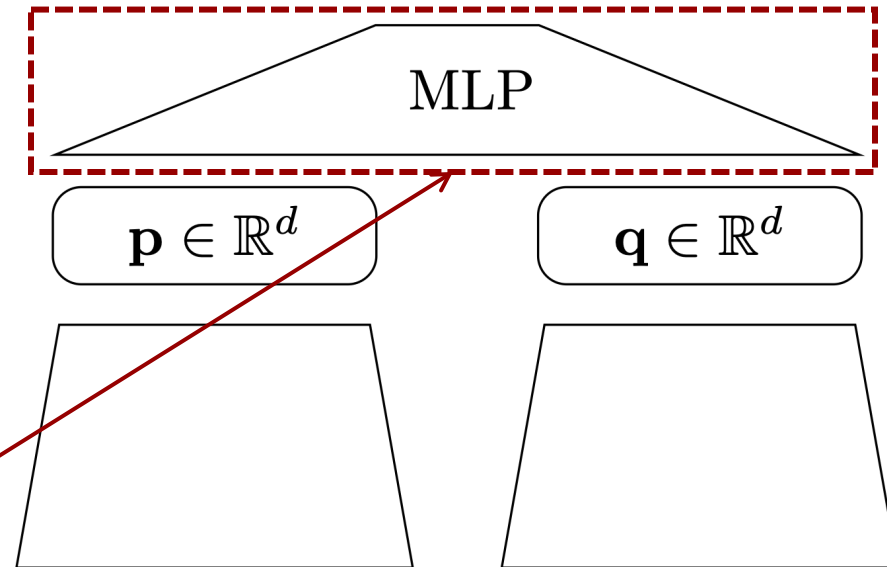
- **Predefined** similarity

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle$$



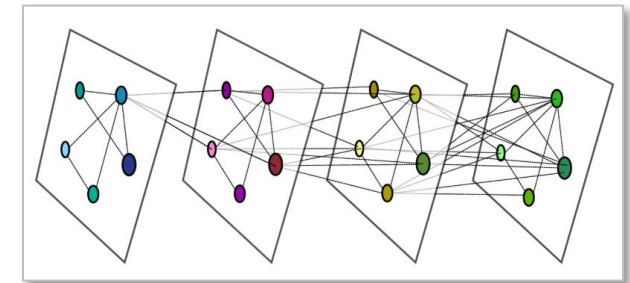
- **Learned** similarity

$$\phi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) = \mathbf{f}_{W_l, \mathbf{b}_l}(\dots \mathbf{f}_{W_1, \mathbf{b}_1}([\mathbf{p}, \mathbf{q}]) \dots)$$



We can make this part even more complicated!

(E.g., graph neural networks)



But is inner product really that bad?

Performance Revisited [Rendle et al., RecSys 2020]

Neural Collaborative Filtering vs. Matrix Factorization Revisited

Steffen Rendle
srendle@google.com
Google Research
Mountain View, California

Li Zhang
liqzhang@google.com
Google Research
Mountain View, California

Walid Krichene
walidk@google.com
Google Research
Mountain View, California

John Anderson
janders@google.com
Google Research
Mountain View, California

ABSTRACT

Embedding based models have been the state of the art in collaborative filtering for over a decade. Traditionally, the dot product or higher order equivalents have been used to combine two or more embeddings, e.g., most notably in matrix factorization. In recent years, it was suggested to replace the dot product with a learned similarity e.g. using a multilayer perceptron (MLP). This approach is often referred to as *neural collaborative filtering* (NCF). In this work, we revisit the experiments of the NCF paper that popularized learned similarities using MLPs. First we show that with a

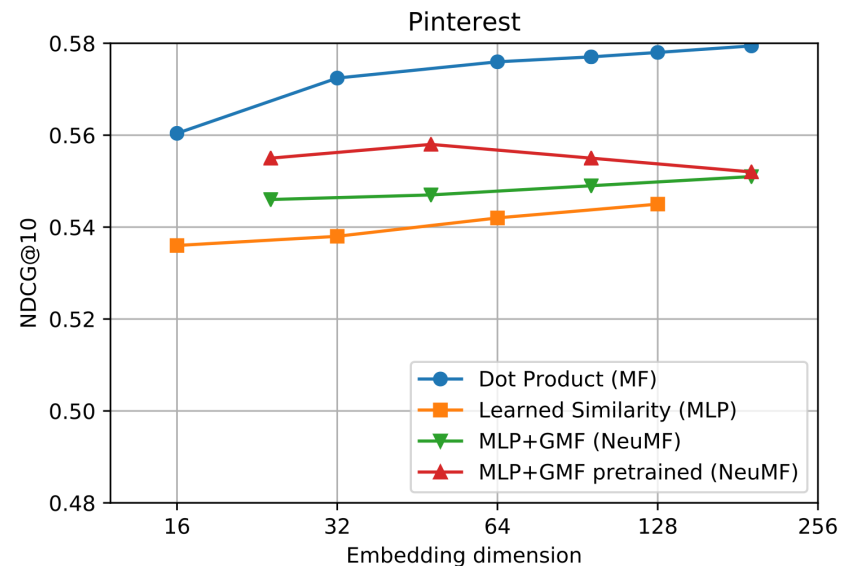
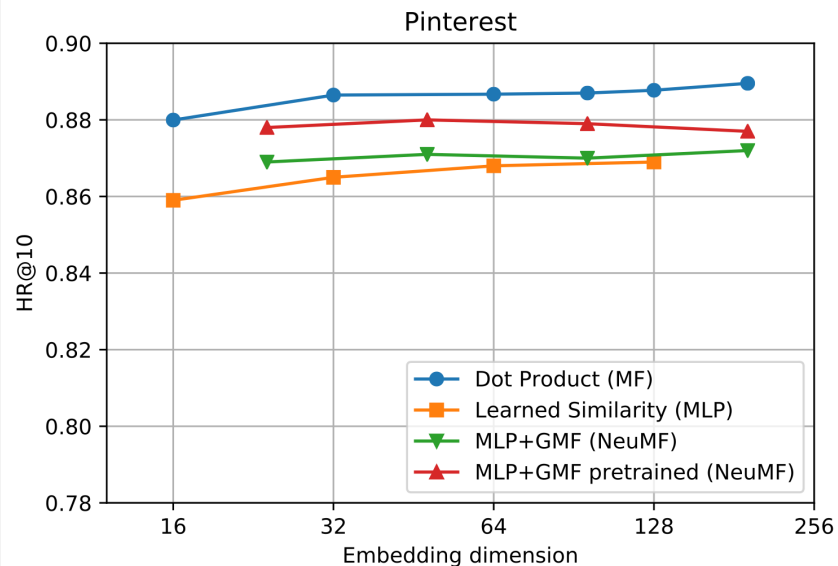
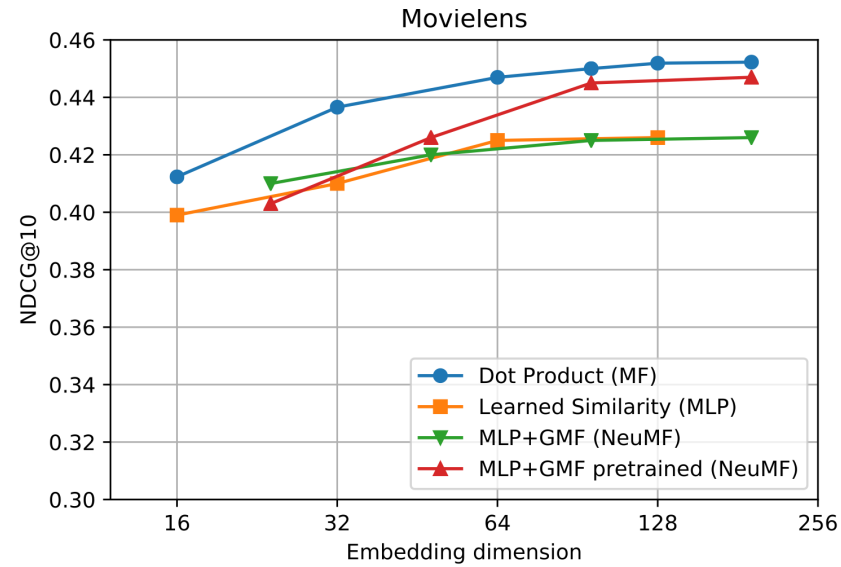
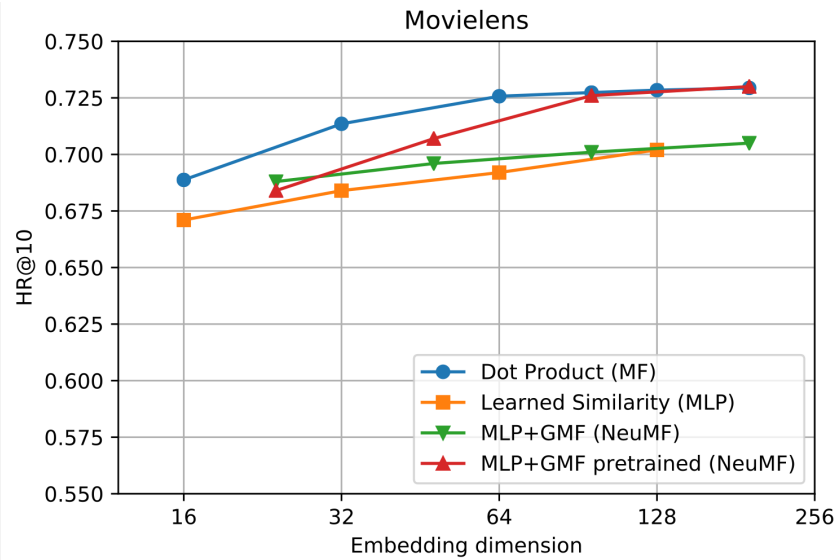
1 INTRODUCTION

Embedding based models have been the state of the art in collaborative filtering for over a decade. A core operation of most of these embedding based models is to combine two or more embeddings. For example, combining a user embedding with an item embedding to obtain a single score that indicates the preference of the user for the item. This can be viewed as a *similarity function* in the embedding space. Traditionally, a dot product or higher order products have been used for the similarity. Recently, it has become popular to learn the similarity function with a neural network. Most

Performance Revisited

- **Motivation:** Learning similarity functions (e.g., **Neural Collaborative Filtering**) has typically been regarded as the de facto standard in academic studies and is considered a strong baseline
- Experimental Setup
 - Follows the Neural Collaborative Filtering paper
 - **Datasets:** MovieLens 1M and Pinterest
 - **Compared Methods:** **Matrix Factorization** (MF) and **Neural Collaborative Filtering** variants

Results

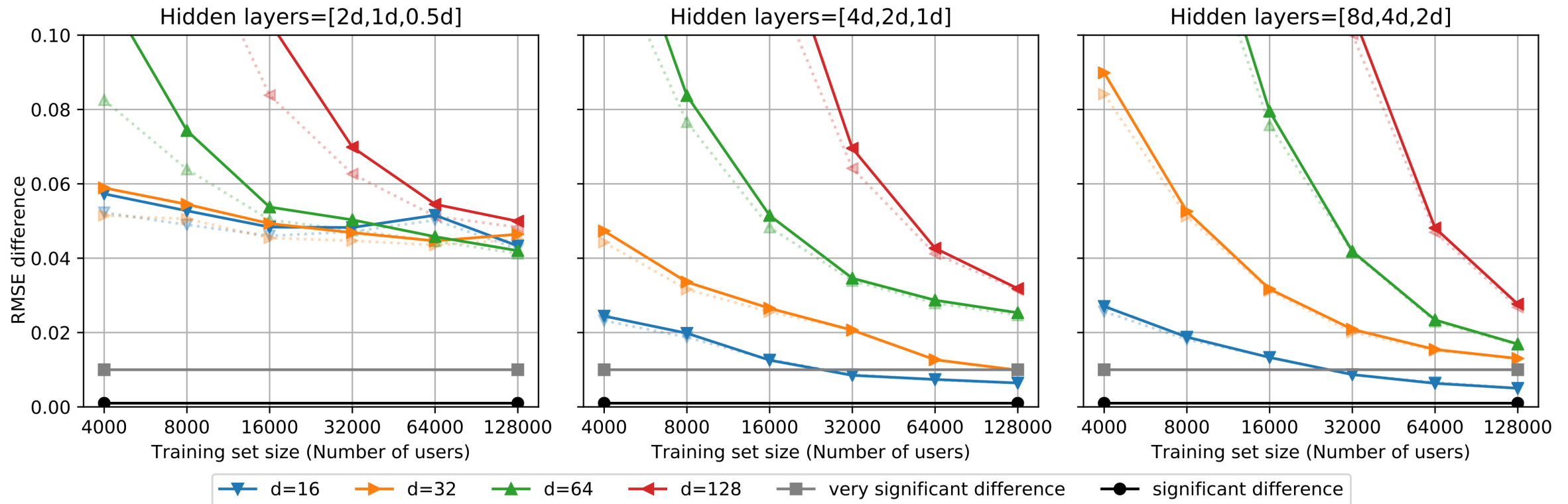


- MF substantially outperforms Neural Collaborative Filtering variants (MLP and NeuMF) on all datasets, evaluation metrics and embedding dimensions.

Why does this happen?

- Why did we believe **Neural Collaborative Filtering** could be better than **MF**?
 - Since **MLPs** can approximate any continuous function under certain conditions (known as the Universal Approximation Theorem), we expect **MLPs** to capture more complex user-item similarity patterns than a simple **inner product**
- But how much training data does **Neural Collaborative Filtering** need to capture complex user-item similarity patterns?
- Let's further simplify the learning problem: how much training data does **Neural Collaborative Filtering** need to capture even the simple **inner product** similarity?

MLPs require massive data even to learn an inner product!

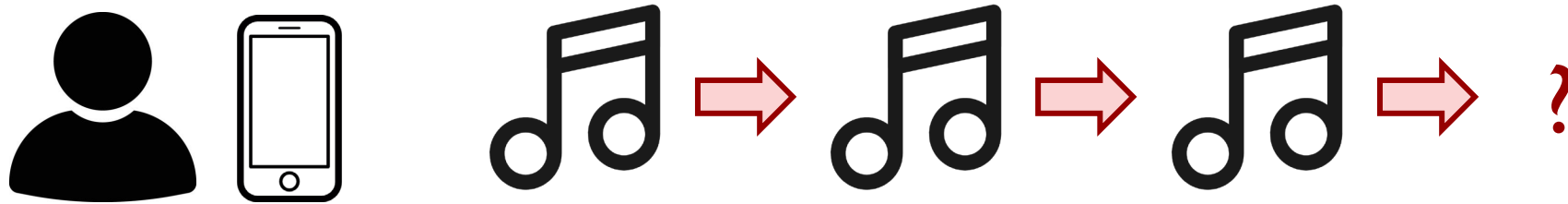


- As models become more complex, we need more training data (for example, the entire Wikipedia and BookCorpus used for BERT). However, in recommender systems, we rarely have access to such large amounts of data.

Neural Recommendation: Sequential Recommendation

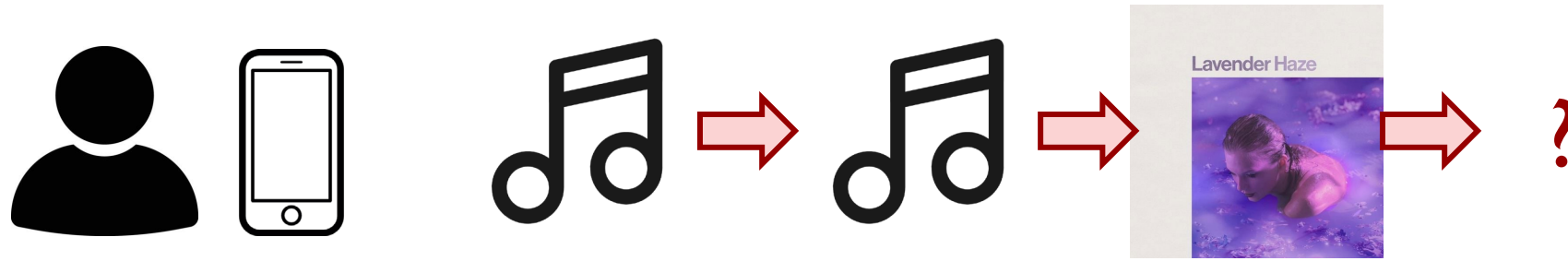
We are adding streaming to our record store!

- What is the next track to stream to a user?



We are adding streaming to our record store!

- What is the next track to stream to a user?



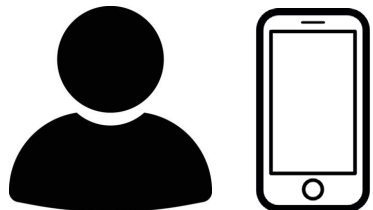
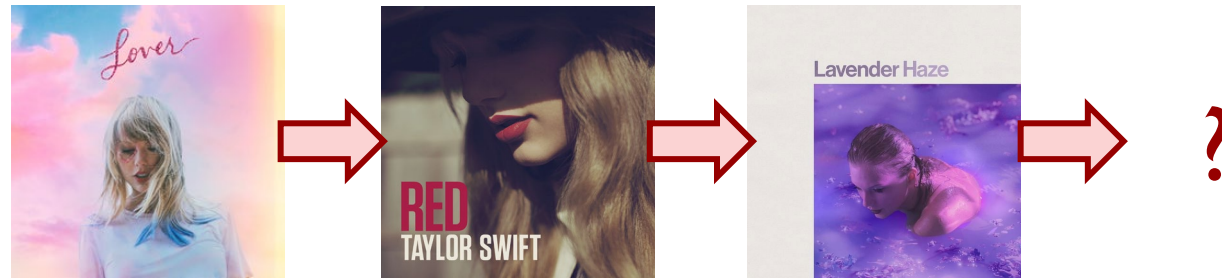
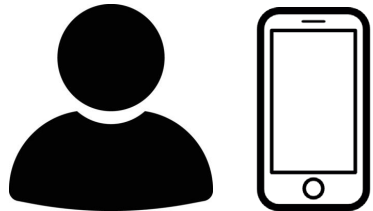
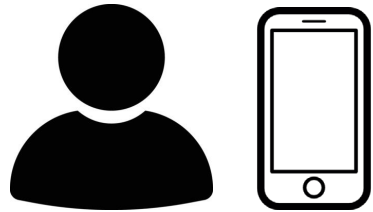
- Our user just listened to Taylor Swift's *Lavender Haze*
 - What should we do next?
 - Play more songs by Taylor Swift?
 - Play more color-themed songs?
 - Play more modern pop songs?
 - Play it again?
- **Question 1:** What do these options have in common?
 - **Question 2:** What other factors should I consider in choosing among these options?

What RecSys techniques have we already learned?

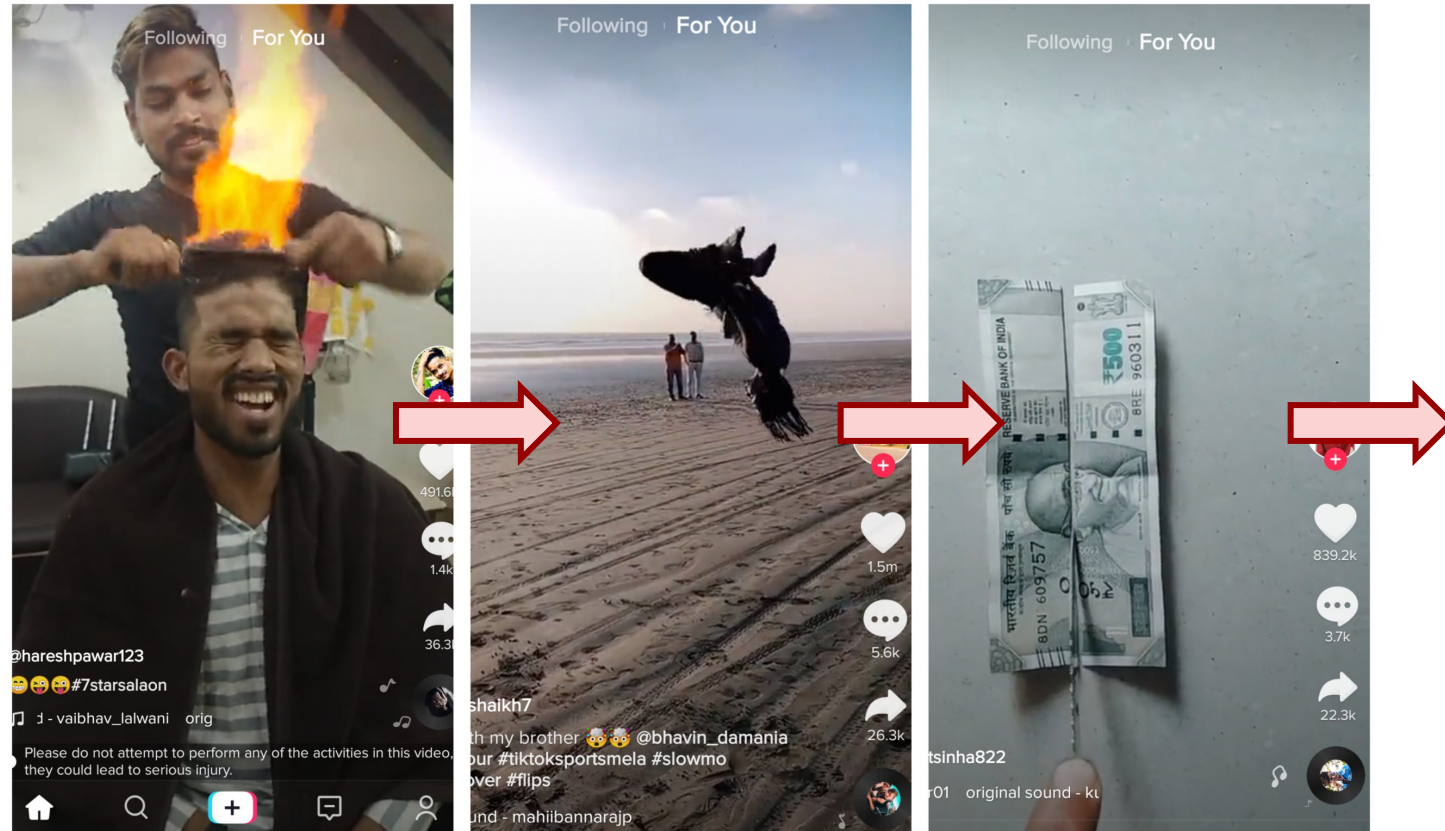
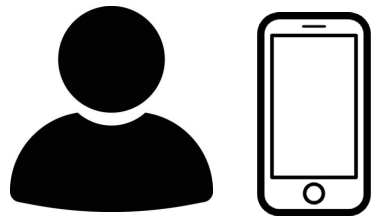
- **Content-Based Approach:** Recommend items that are **similar to** those previously viewed
 - Here, the definition of **similarity** is based on **attributes** (e.g., lyrics)
- **Item-Item Collaborative Filtering:** Given a candidate item, first identify its **similar** items, and then check whether the user has viewed any of them
 - Here, the definition of **similarity** is based on the **columns corresponding to the two items** in the user-item matrix
- **Matrix Factorization:** Find the item in the **embedding space** that are **closest to** the user and have not yet been viewed
 - A user's **embedding** should be **close to** the **embeddings** of the items they have already viewed
- **Neural Collaborative Filtering:** More sophisticated **similarity** computation methods, but still based on **embeddings**
- **Summary:** Find items **similar to** those a user has already viewed. The only difference lies in how **similarity** is defined

What are we missing?

- Understanding dynamic user intent from sequential interactions



Other Sequential RecSys Examples: TikTok



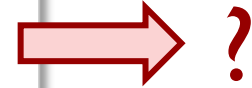
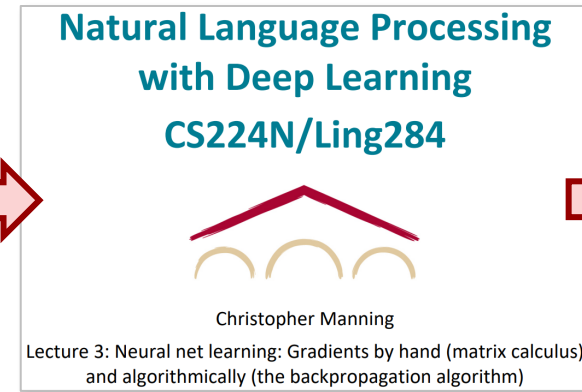
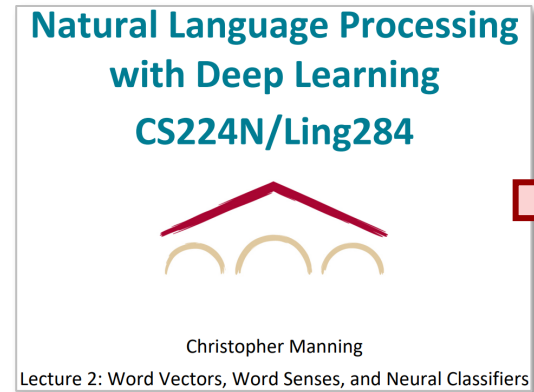
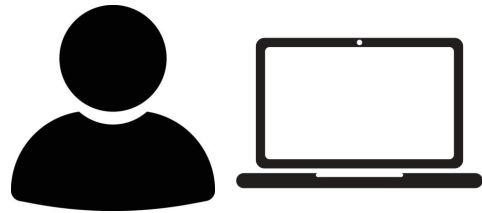
?

Other Sequential RecSys Examples: Netflix



- For certain products, the **order** can be crucial!
 - [movies and books] Suppose I just watched *Harry Potter 2*. Would it make sense to recommend *Harry Potter 1*?
 - [lecture videos] Suppose I just watched the video for **Lecture 3** of Stanford's CS224N. Would it make sense to recommend the video for **Lecture 2**?
 - [electronic devices] Suppose I just bought an **iPhone XR**. Would it make sense to recommend an **iPhone 8**?

Other Sequential RecSys Examples: YouTube and Amazon



What are we missing?

- Embedding similarity is **symmetric**.
 - If the user has just viewed item i , whether the model will recommend item j as the next item fully depends on $\text{score}(i, j) = \mathbf{e}_i^T \mathbf{e}_j$ or $\cos(\mathbf{e}_i, \mathbf{e}_j)$
 - If the user has just viewed item j , whether the model will recommend item i as the next item fully depends on $\text{score}(j, i) = \text{score}(i, j)$
- Sequential recommendation is **asymmetric**.
 - When your model recommends item j as the next item after item i , it does NOT necessarily mean that i should be recommended as the next item after j .
 - We should model something like $\text{score}(j | i) \neq \text{score}(i | j)$

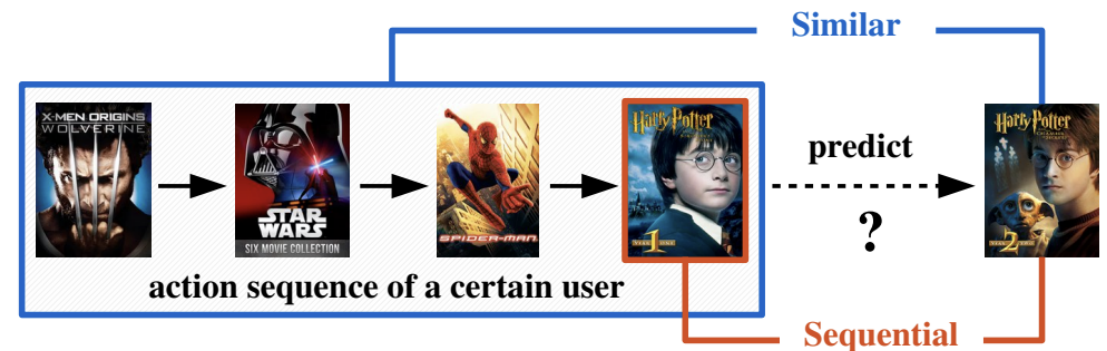
Transition matrix is asymmetric!

Fossil [He and McAuley, ICDM 2016]

Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation

Ruining He, Julian McAuley
Department of Computer Science and Engineering
University of California, San Diego
Email: {r4he, jmcauley}@cs.ucsd.edu

Abstract—Predicting personalized sequential behavior is a key task for recommender systems. In order to predict user actions such as the next product to purchase, movie to watch, or place to visit, it is essential to take into account both long-term user preferences and sequential patterns (i.e., short-term dynamics). Matrix Factorization and Markov Chain methods have emerged as two separate but powerful paradigms for modeling the two respectively. Combining these ideas has led





Rucanor Aluminium Baseball Bat

by [Rucanor](#)

★★★★☆ (29 customer reviews)

RRP: ~~£24.00~~

Price: **£18.40** & **FREE Delivery** in the UK. [Details](#)

You Save: **£2.60** (12%)

Colour: Silver

Size: 60

In stock.

Sold by [MA sports uk](#) and [Fulfilled by Amazon](#). Gift-wrap available.

- 60 CM BASEBALL BAT
- ALUMINIUM Alloy

5 new from **£14.40**

Quantity: 1 ▼

[Add to Basket](#)

or 1-Click Checkout

[Buy now with 1-Click®](#)

☐ This is a gift

[Add to Wish List](#) ▼

More Buying Choices

ANDRA SPORTS [Add to Basket](#)
£14.40 + £3.99 UK delivery

sportknox [Add to Basket](#)
£17.45 + £3.99 UK delivery

5 new from **£14.40**

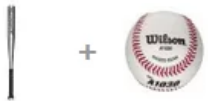
Have one to sell? [Sell yours here](#)

[Share](#) [Email](#) [Facebook](#) [Twitter](#) [Pinterest](#)

Roll over image to zoom in

[Share your own customer images](#)

Frequently Bought Together



Price For Both: **£28.39**

[Add both to Basket](#)

[Show availability and delivery details](#)

- ☒ **This item:** Rucanor Aluminium Baseball Bat, Silver - 60 cm **£18.40**
- ☒ WILSON Official League Individual Baseball **£9.99**

Customers Who Bought This Item Also Bought

Page 1 of 18



REYDON Softball Ball
★★★★★ (1)
£3.99



MFH Balaclava 3 Hole Black
★★★★☆ (94)
£1.74



New Midwest Slugger
Baseball Glove Vinyl
Catching Mitt Left Hand
Junior / Senior
★★★★★ (5)



Rawlings 9" Indoor / Outdoor
T-Ball Training Baseball -
TVB
★★★★★ (3)
★★★★★ (5)

- Predict a user's next step based on the next-step behavior patterns of the **overall population**
- Amazon's "*Customer Who Bought This Item Also Bought*"

Markov Chain to Model Sequential Dynamics

- Historical user-item interactions

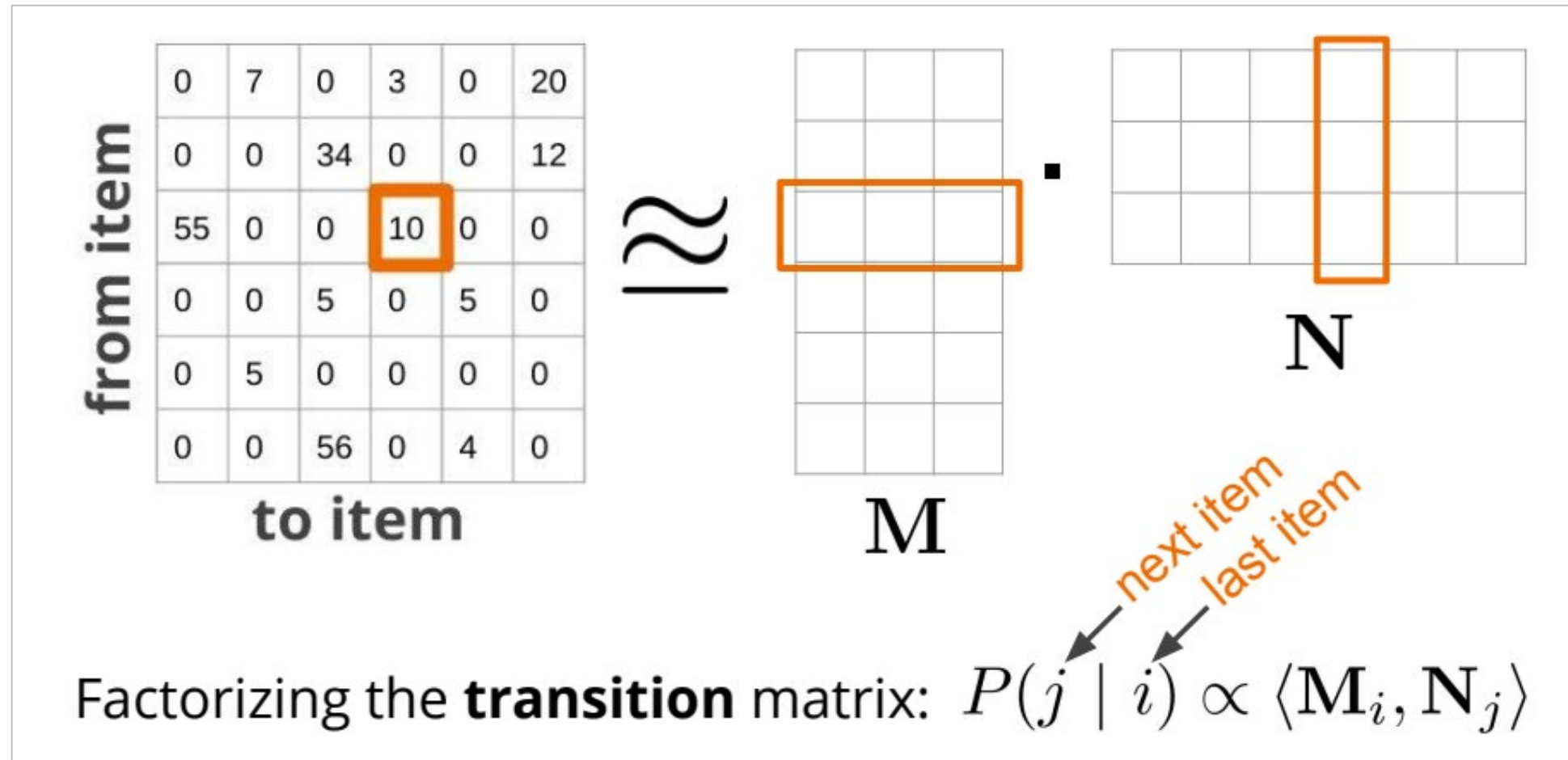
User								
<i>u1</i>	→	<i>i1</i>	→	<i>i2</i>	→	<i>i4</i>		
<i>u2</i>	→	<i>i3</i>	→	<i>i4</i>				
<i>u3</i>	→	<i>i1</i>	→	<i>i2</i>	→	<i>i4</i>	→	<i>i3</i>
<i>u4</i>	→	<i>i3</i>	→	<i>i1</i>	→	<i>i2</i>		

- Step 1:** Construct a transition matrix

	<i>i1</i>	<i>i2</i>	<i>i3</i>	<i>i4</i>
<i>i1</i>	0	3	0	0
<i>i2</i>	0	0	0	2
<i>i3</i>	1	0	0	1
<i>i4</i>	0	0	1	0

Markov Chain to Model Sequential Dynamics

- **Step 2:** Factorize the transition matrix

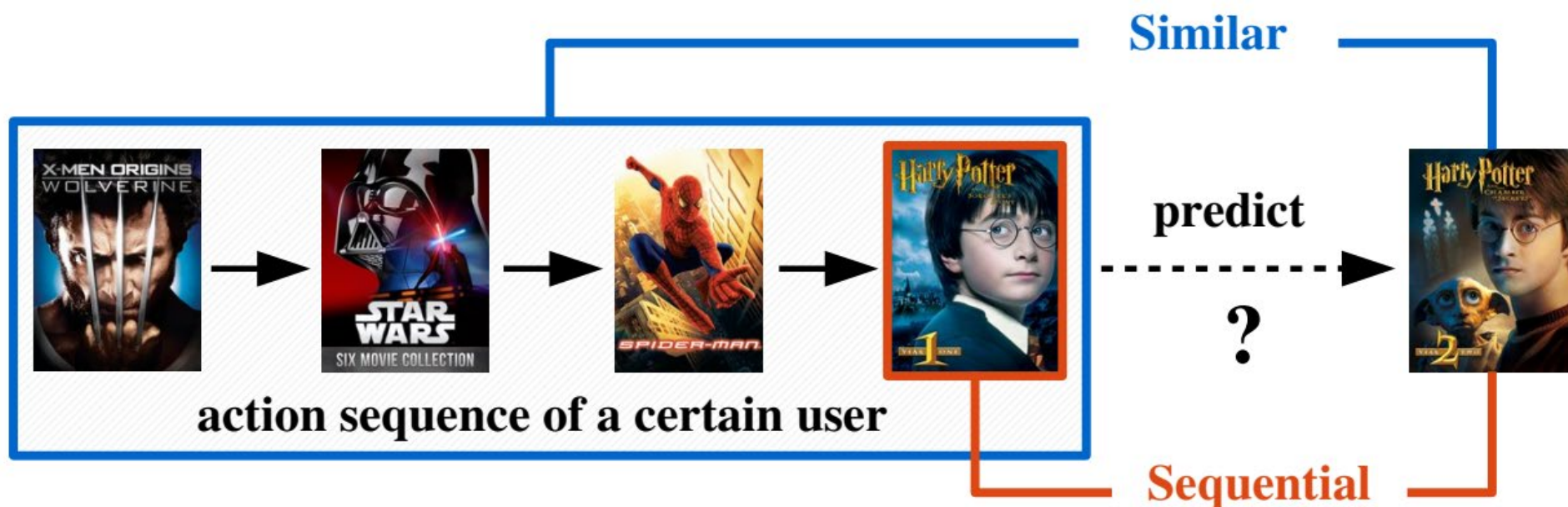


Markov Chain to Model Sequential Dynamics

- Problems with this approach?
 - NOT personalized!
 - Regardless of which user the model is considering, $\text{score}(j \mid i) \propto \mathbf{M}_i^T \mathbf{N}_j$
 - But $\mathbf{M}_i$ and $\mathbf{N}_j$ are the same for all users
- How to make the model personalized?

$$p_u(j \mid i) \propto \underbrace{\sum_{j' \in \mathcal{I}_u^+ \setminus \{j\}} \langle \mathbf{P}_{j'}, \mathbf{Q}_j \rangle}_{\text{user preferences}} + \underbrace{(\eta + \eta_u)}_{\text{personalized weighting factor}} \cdot \underbrace{\langle \mathbf{M}_i, \mathbf{N}_j \rangle}_{\text{sequential dynamics}}$$

$$p_u(j \mid i) \propto \underbrace{\sum_{j' \in \mathcal{I}_u^+ \setminus \{j\}} \langle \mathbf{P}_{j'}, \mathbf{Q}_j \rangle}_{\text{user preferences}} + \underbrace{(\eta + \eta_u)}_{\text{personalized weighting factor}} \cdot \underbrace{\langle \mathbf{M}_i, \mathbf{N}_j \rangle}_{\text{sequential dynamics}}$$



Back to Motivating Example

- Play *Lavender Haze* again? Play more songs by Taylor Swift? Play more color-themed songs?
 - Might all be likely according to the transition matrix. But if you consider user preference ...



Higher-Order Markov Chain

User								
$u1$	→	$i1$	→	$i2$	→	$i4$		
$u2$	→	$i3$	→	$i4$				
$u3$	→	$i1$	→	$i2$	→	$i4$	→	$i3$
$u4$	→	$i3$	→	$i1$	→	$i2$		

- Transition matrix:

	$i1$	$i2$	$i3$	$i4$
$i1 \rightarrow i2$	0	0	0	2
$i1 \rightarrow i3$	0	0	0	0
$i1 \rightarrow i4$	0	0	0	0
$i2 \rightarrow i1$	0	0	0	0
...	...	...	...	...

- **Sparse**
- # of rows **increases exponentially** with the order of the Markov chain
- Similar to the n-gram language model
- Can we use a **neural language model** to alleviate sparsity?

Performance

- Fossil outperforms BPR-MF (and other static baselines) because it models sequential dynamics
- Small orders seem to be enough to achieve good performance

Table IV: AUC on different datasets (higher is better). The number of latent dimensions for all comparison methods (except for POP) is set to $K = 10$. For *Fossil*, we test different orders of the Markov Chain (i.e., 1, 2, and 3). On the right we demonstrate the improvement of FPMC vs. BPR-MF, *Fossil* vs. FISM, *Fossil* vs. FPMC, and *Fossil* vs. the best baselines.

Dataset	(a) POP	(b) BPR-MF	(c) FISM	(d) FMC	(e) FPMC	(f-1) <i>Fossil</i>	(f-2) <i>Fossil</i>	(f-3) <i>Fossil</i>	% improvement			
									e vs. b	f vs. c	f vs. e	f vs. best
<i>Amazon-Office</i>	0.6427	0.6736	0.7113	0.6874	0.6891	0.7211	0.7224	0.7221	2.30%	1.56%	4.83%	1.56%
<i>Amazon-Auto</i>	0.5870	0.6379	0.6736	0.6452	0.6446	0.6910	0.6904	0.6901	1.05%	2.58%	7.20%	2.58%
<i>Amazon-Game</i>	0.7495	0.8483	0.8639	0.8401	0.8502	0.8793	0.8813	0.8817	0.22%	2.06%	3.71%	2.06%
<i>Amazon-Toy</i>	0.6240	0.7020	0.7499	0.6665	0.7061	0.7625	0.7645	0.7652	0.58%	2.04%	8.37%	2.04%
<i>Amazon-Cell</i>	0.6959	0.7212	0.7755	0.7359	0.7396	0.7982	0.8009	0.8006	2.55%	3.27%	8.29%	3.27%
<i>Amazon-Clothes</i>	0.6189	0.6513	0.7085	0.6673	0.6672	0.7255	0.7256	0.7259	2.44%	2.46%	8.80%	2.46%
<i>Amazon-Elec</i>	0.7837	0.7927	0.8210	0.7992	0.7985	0.8411	0.8438	0.8444	0.73%	2.85%	5.75%	2.85%
<i>Epinions</i>	0.4576	0.5520	0.5818	0.5532	0.5477	0.6014	0.6050	0.6048	-0.78%	3.99%	10.46%	3.99%
<i>Foursquare</i>	0.9168	0.9506	0.9230	0.9441	0.9485	0.9626	0.9621	0.9618	-0.22%	4.29%	1.49%	1.26%
Avg. ($K = 10$)	0.6751	0.7255	0.7565	0.7265	0.7324	0.7759	0.7773	0.7774	0.99%	2.79%	6.54%	2.45%
Avg. ($K = 20$)	0.6751	0.7285	0.7580	0.7293	0.7344	0.7780	0.7795	0.7788	0.88%	2.83%	6.44%	2.54%



Thank You!

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F26.html>