Are You My Neighbor? Bringing Order to Neighbor Computing Problems.

David C. Anastasiu[1,2] , Huzefa Rangwala[3], and Andrea Tagarelli[4]

[1]Computer Engineering, San Jose State University, CA

[1]Computer Science & Engineering, Santa Clara University, CA

[2]Computer Science & Engineering, George Mason University, VA

[3]DIMES, University of Calabria, Italy

25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 2019

Part IV: Neighbors in Advertising and Recommender Systems

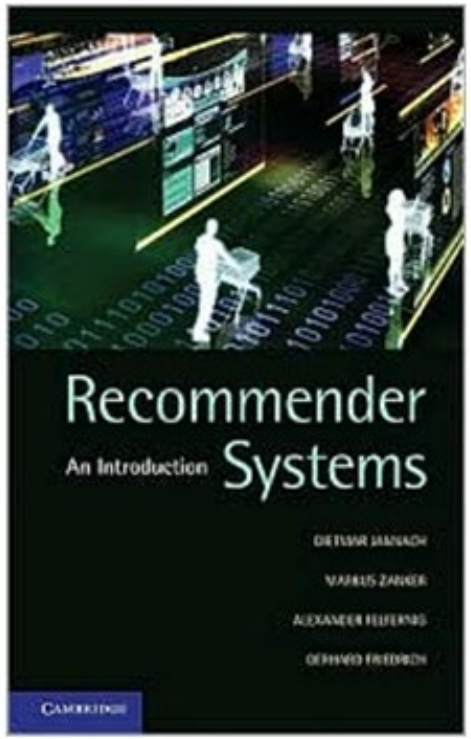
Huzefa Rangwala, George Mason University [rangwala@cs.gmu.edu]
David C. Anastasiu, San José State University [david.anastasiu@sjsu.edu]

Tutorial Outline

- **Part I: Problems and Data Types**
  - Dense, sparse, and asymmetric data
  - Bounded nearest neighbor search
  - Nearest neighbor graph construction
  - Classical approaches and limitations
- **Part II: Neighbors in Genomics, Proteomics, and Bioinformatics**
  - Mass spectrometry search
  - Microbiome analysis
- **Part III: Approximate Search**
  - Locality sensitive hashing variants
  - Permutation and graph-based search
  - Maximum inner product search
- **Part IV: Neighbors in Advertising and Recommender Systems**
  - Collaborative filtering at scale
  - Learning models based on the neighborhood structure
- **Part V: Filtering-Based Search**
  - Massive search space pruning by partial indexing
  - Effective proximity bounds and when they are most useful
- **Part VI: Neighbors in Learning and Mining Problems in Graph Data**
  - Neighborhood as cluster in a complex network system
  - Neighborhood as influence trigger set

Collaborative Filtering and Neighborhoods in Recommender Systems

Huzefa Rangwala, George Mason University [rangwala@cs.gmu.edu]



Recommender Systems: An Introduction

by Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich

AVERAGE CUSTOMER RATING:

(Be the first to review)

Gefällt mir Registrieren, um sehen zu können, was deinen Freunden gefällt.

FORMAT:
Hardcover

NOOKbook (eBook) - not available

Tell the publisher you want this in NOOKbook format

NEW FROM BN.COM

~~$65.00~~ List Price

$52.00 Online Price

(You Save 20%)

Add to Cart

NEW & USED FROM OUR

New starting at $56.46(You S

Used starting at $51.98(You S

See All Prices

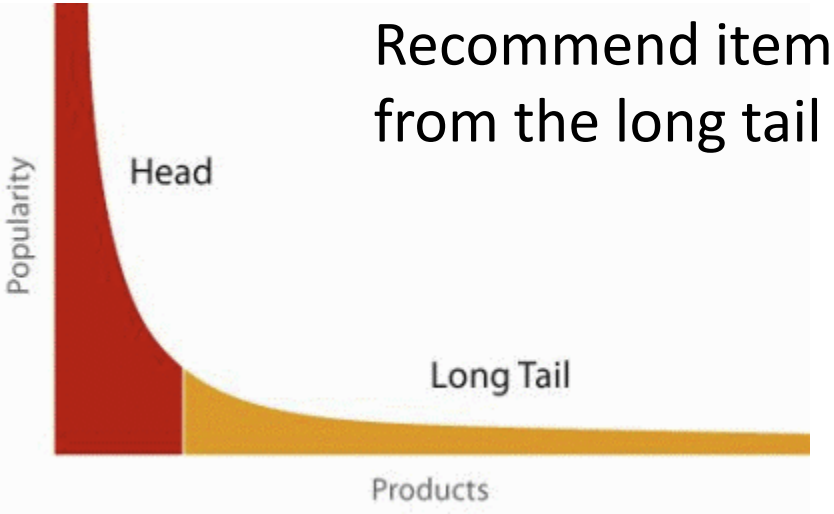
Table of Contents

Customers who bought this also bought

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.
- Finally:
  - Recommend items that are assumed to be relevant
- But:
  - Remember that relevance might be context-dependent
  - Characteristics of the list itself might be important (diversity)

Where does a RS do its job well?



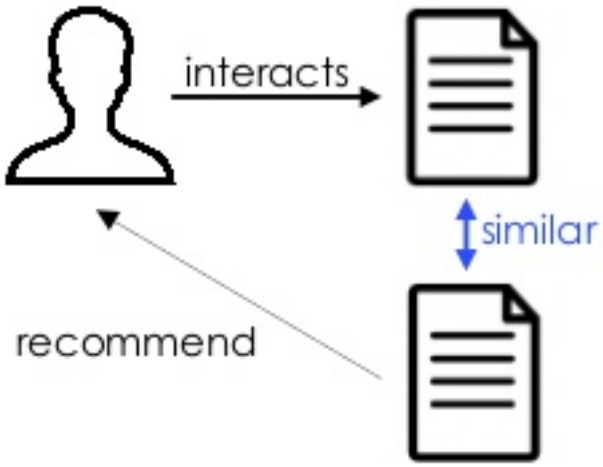
Recommend items from the long tail

- “Recommend widely unknown items that users might actually like!”.
- 20% of items accumulate 74% of all positive ratings.

Items rated > 3 in MovieLens 100K dataset.

Recommender System Types

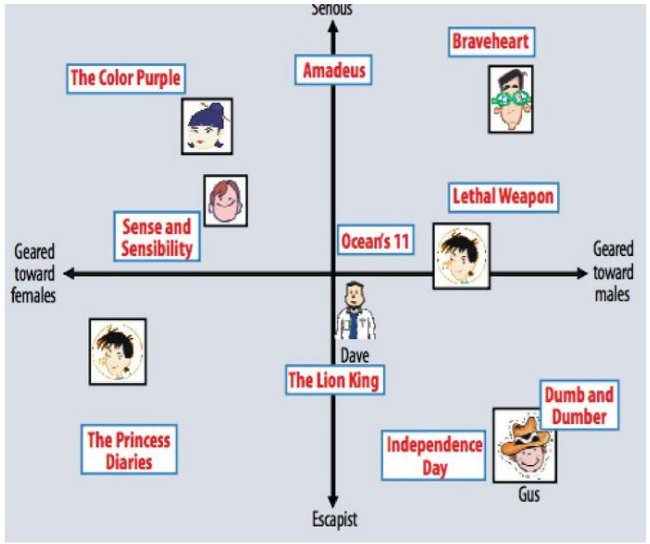
CONTENT BASED FILTERING



Recommends users similar items that the user has liked in the past

COLLABORATIVE FILTERING

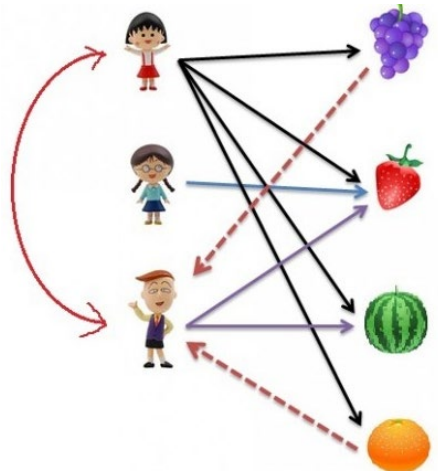
MODEL BASED



Recommend a movie based on its proximity to the user in latent space

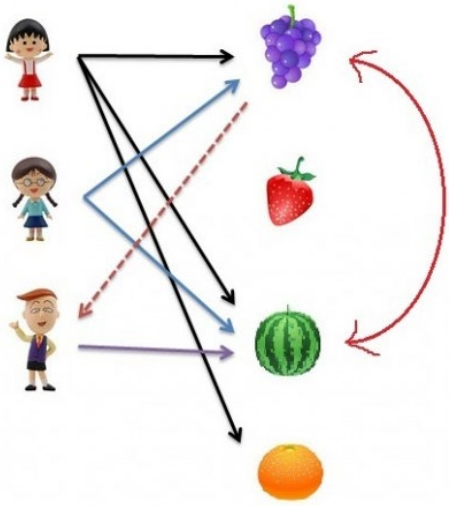
MEMORY BASED

USER BASED



"Users who are similar to you also liked …"

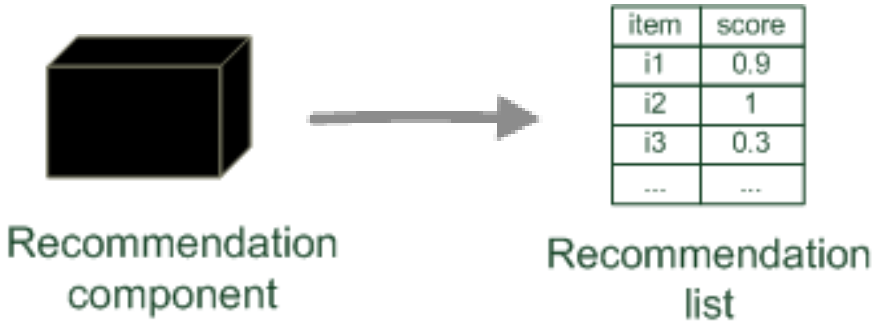
ITEM BASED



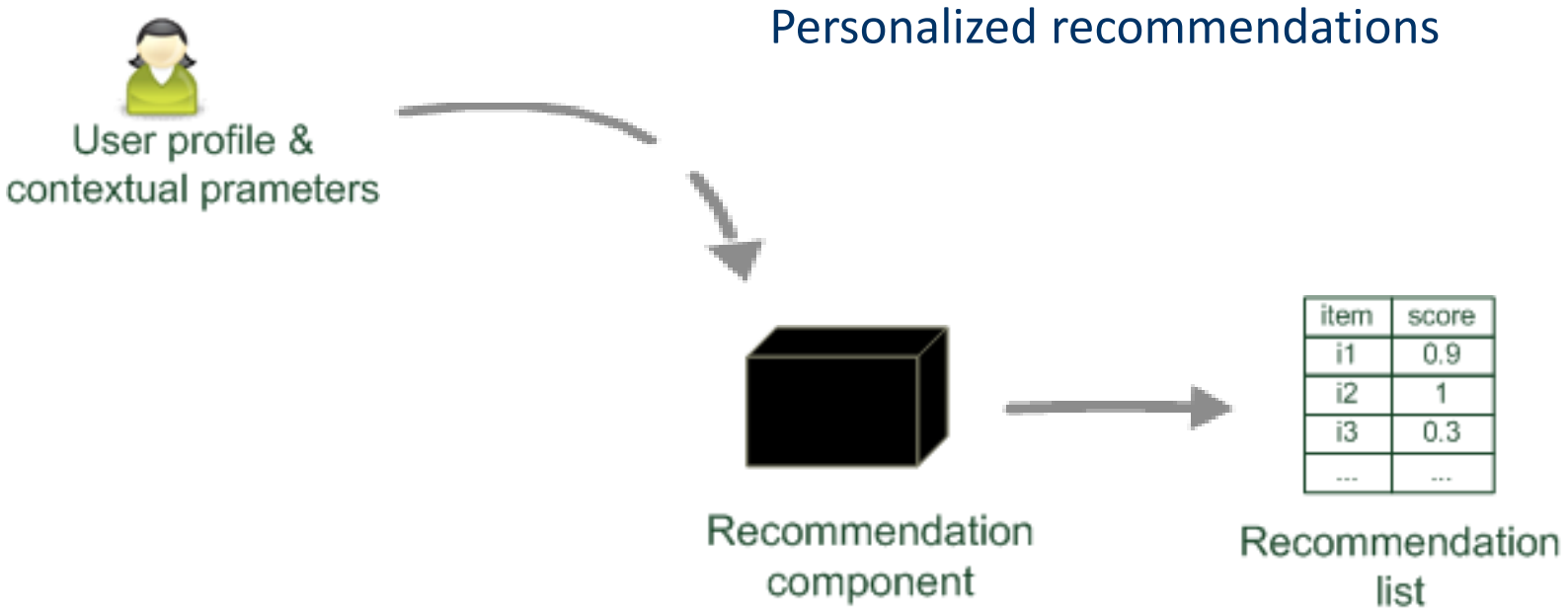
"Users who liked this item also liked.."

Paradigms of recommender systems

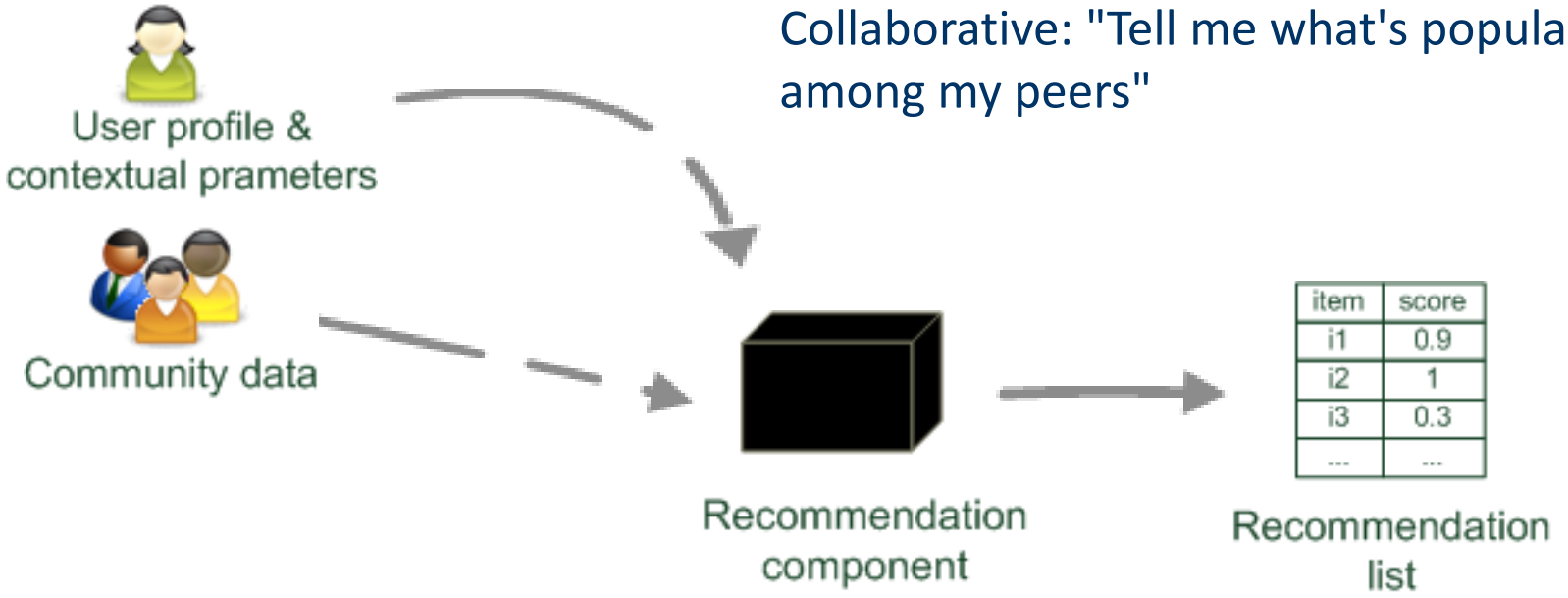
Recommender systems reduce information overload by estimating relevance



Paradigms of recommender systems



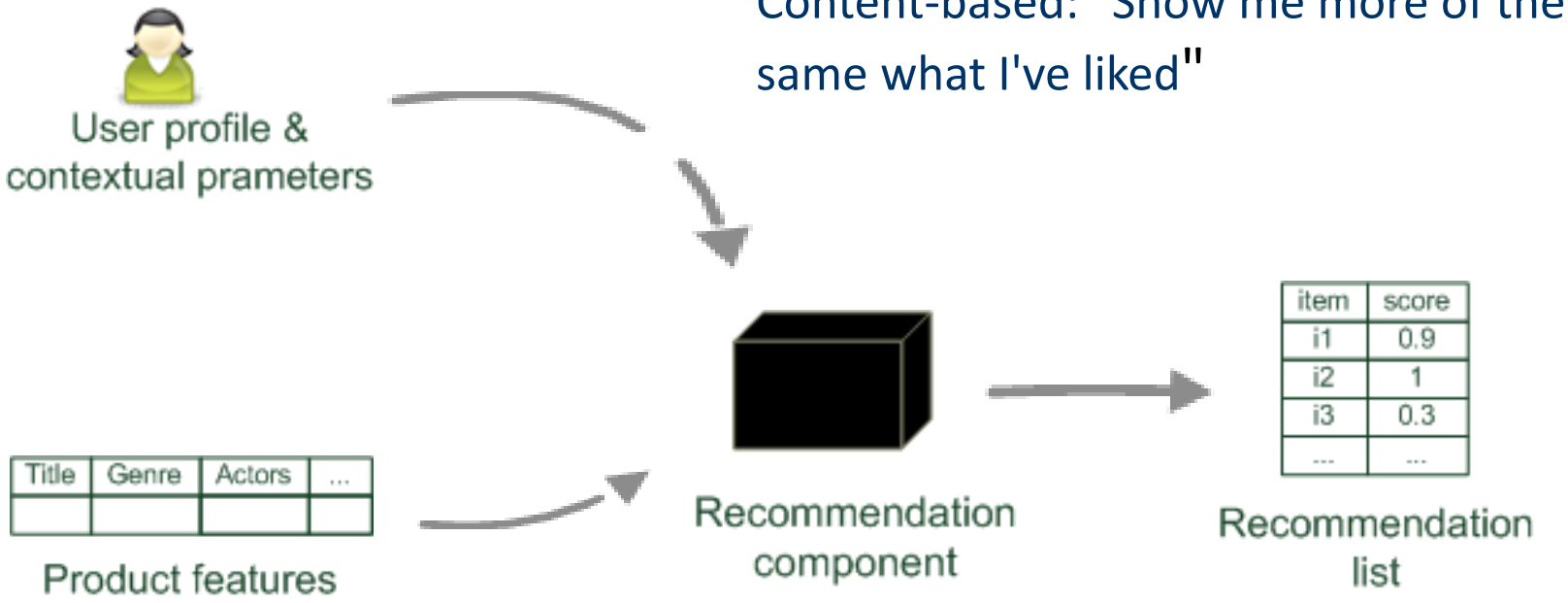
Paradigms of recommender systems



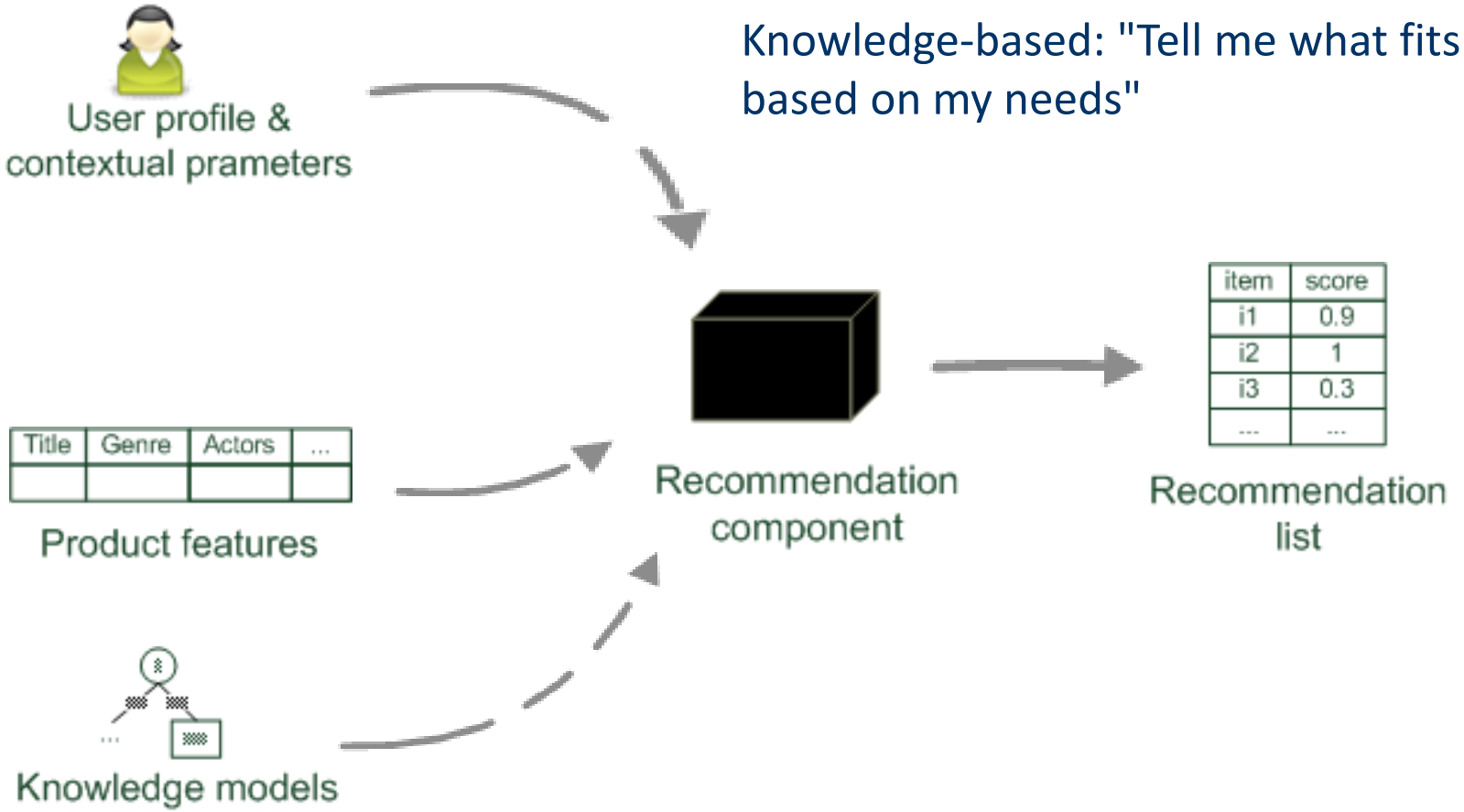
Collaborative: "Tell me what's popular among my peers"

Paradigms of recommender systems

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

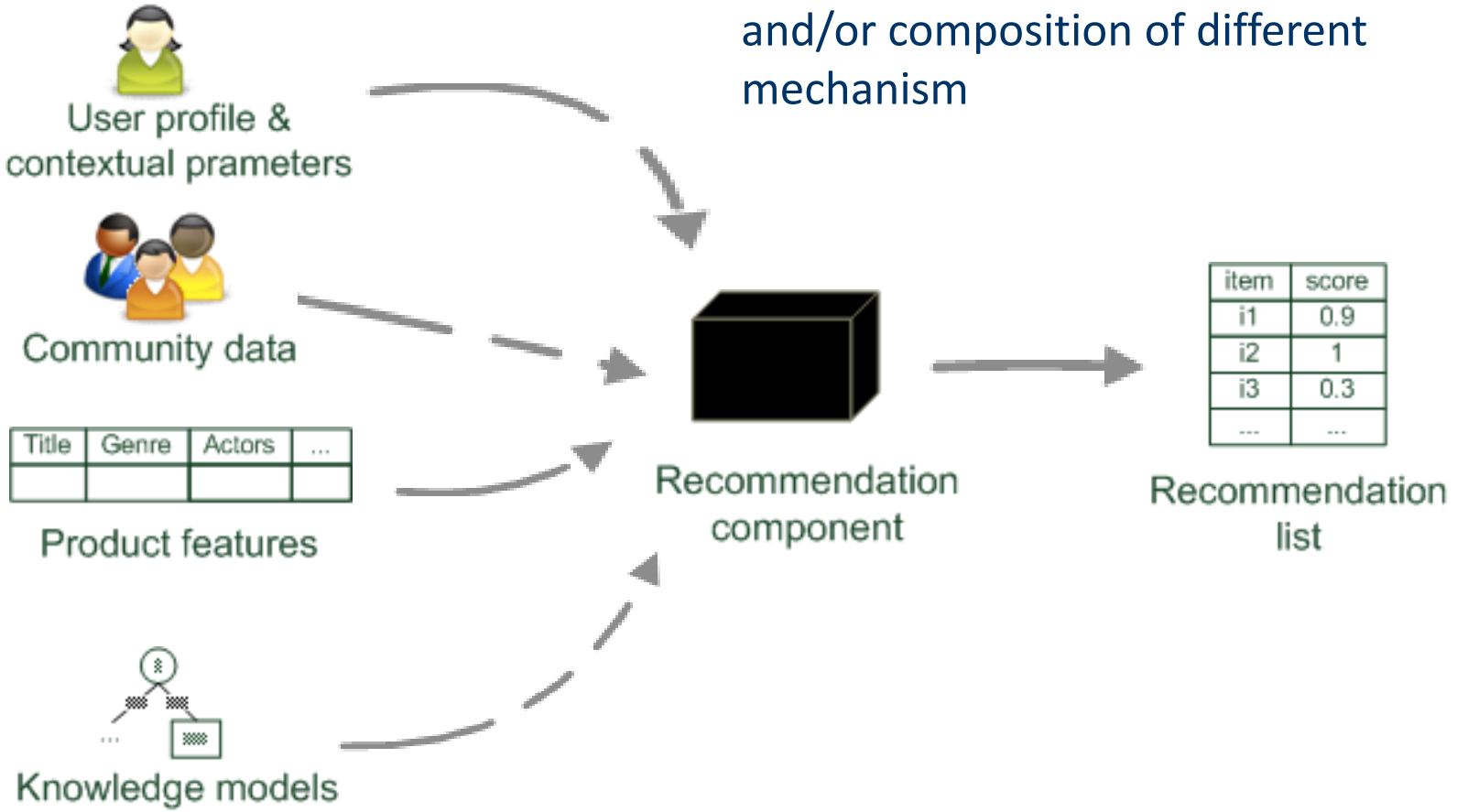


Paradigms of recommender systems



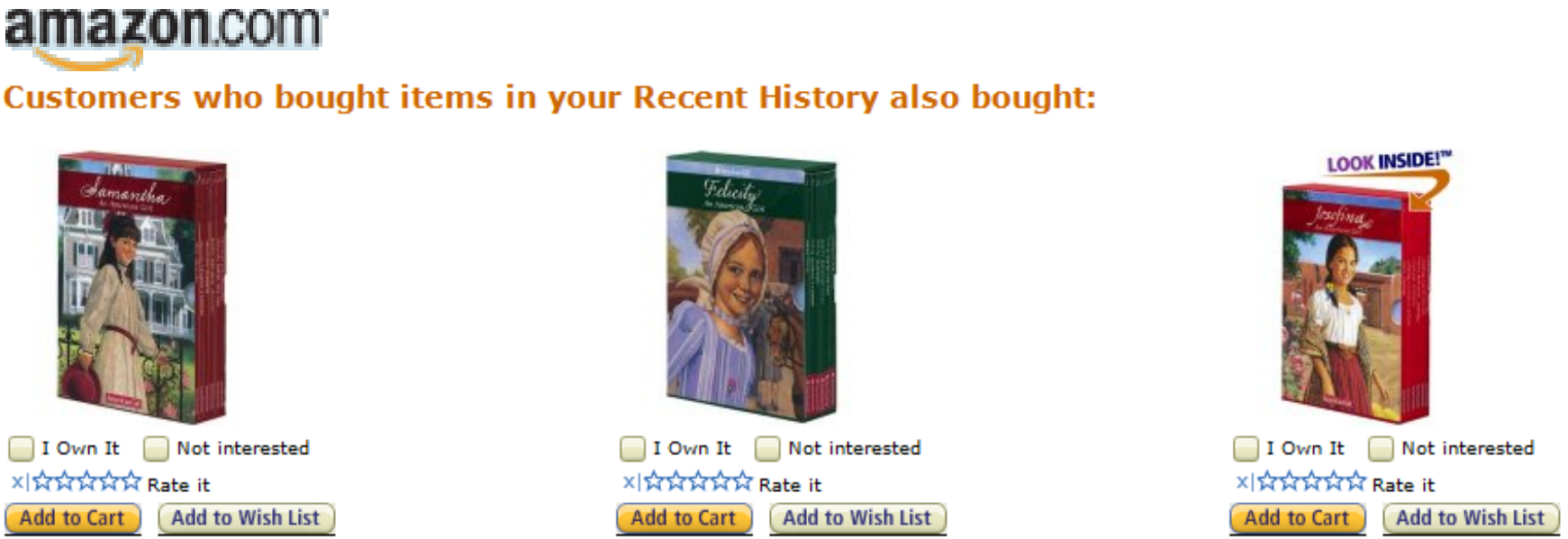
Paradigms of recommender systems

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



Collaborative filtering (CF)

- Recommend items based on past transactions of users
- Analyze relations between users and/or items
- Specific data characteristics are irrelevant
  - Domain-free: user/item attributes are not necessary
  - Can identify elusive aspects
- Basic assumption
  - Customer preferences remain the same over time



User-based nearest-neighbor collaborative filtering

- The basic technique:
  - Given an "active user" (Alice) and an item I not yet seen by Alice
  - The *goal is to estimate Alice's rating for this item*, e.g., by
    - find a set of users (peers) 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

	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

User-based nearest-neighbor collaborative filtering

- 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

Pearson correlation

users

$$\mathrm{PC}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

The subset of items that were co-rated by both users

items

$$\mathrm{PC}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

The subset of users that co-rated by both items

- Most widely used for rating prediction for both user- and item-based methods.
- Cosine similarity also works well for item-based top N recommendation.

Making predictions

- A common prediction function:

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

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

Making recommendations

- Making predictions is typically not the ultimate goal
- Usual approach (in academia)
  - Rank items based on their predicted ratings
- However
  - This might lead to the inclusion of (only) niche items
  - **In practice also:** Take item popularity into account
- Approaches
  - "Learning to rank"
    - Optimize according to a given rank evaluation metric (see later)

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
- Value of number of co-rated items
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- Case amplification
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors

Item-based collaborative filtering recommendation algorithms, B. Sarwar et al., WWW 2001

- Scalability issues arise with U2U if many more users than items (m >> n , m = |users|, n = |items|)
  - e.g. Amazon.com
  - Space complexity $O(m^2)$ when pre-computed
  - Time complexity for computing Pearson $O(m^2n)$
- High sparsity leads to few common ratings between two users
- Basic idea: "Item-based CF exploits relationships between items first, instead of relationships between users"

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	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

Data sparsity problems

- Cold start problem
  - How to recommend new items? What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- Alternatives
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be to small to make good predictions
    - Assume "transitivity" of neighborhoods

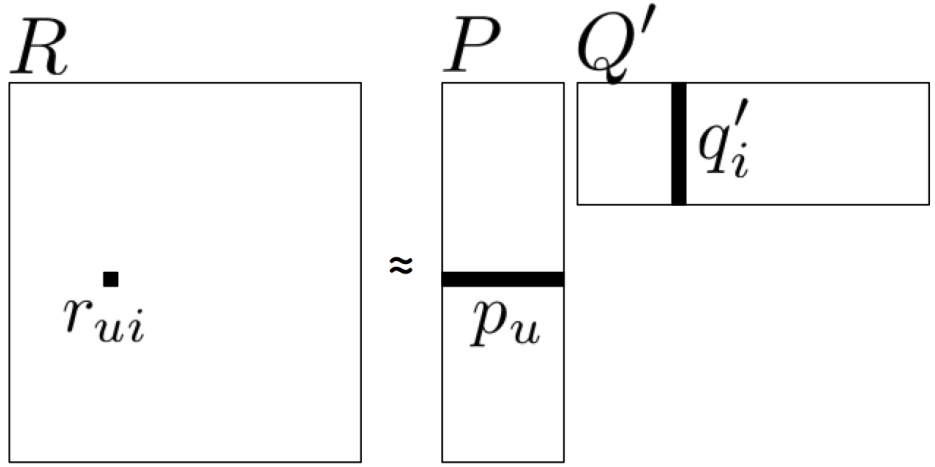
Model-based approaches

- Plethora of different techniques proposed in the last years, 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?

- Basic idea: Trade more complex offline model building for faster online prediction generation
- Singular Value Decomposition for dimensionality reduction of rating matrices
  - Captures important factors/aspects and their weights in the data
  - factors can be genre, actors but also non-understandable ones
  - Assumption that k dimensions capture the signals and filter out noise (K = 20 to 100)
- Constant time to make recommendations
- Approach also popular in IR (Latent Semantic Indexing), data compression, …

Low-rank matrix approximation

- The ranking matrix can be approximated as the product of two low-rank matrices



- The low-rank decomposition can be used to compute the ratings for unseen items as:

$$\hat{r}_{ui} = p_u q_i'$$

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**

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in K} (\hat{r}_{ui} - r_{ui})^2}{|K|}}$$

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

- **Merges neighborhood models with latent factor models**
- **Latent factor models**
  - good to capture weak signals in the overall data
- **Neighborhood models**
  - good at detecting strong relationships between close items
- **Combination in one prediction single function**
  - Local search method such as stochastic gradient descent to determine parameters
  - Add penalty for high values to avoid over-fitting

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$$

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

SLIM – Sparse Linear Model

- Extension of item-based neighbor schemes
  - Model is the pre-computed item-item similarity matrix S

$$\hat{r}_{ui} = r_u s'_i, \text{ where, } R \in \mathcal{R}^{n \times m}, S \in \mathcal{R}^{m \times m}$$

  - Can S be learned in a different way?

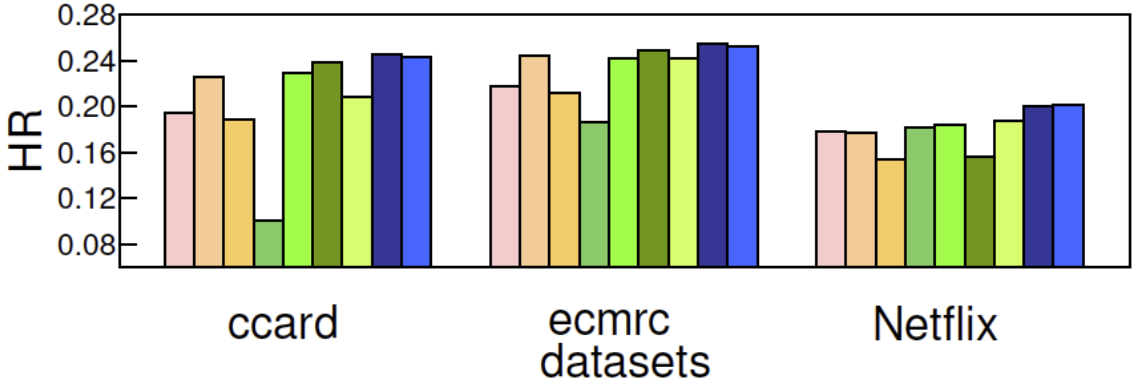
$$\begin{aligned} \min_{S} \quad & \frac{1}{2}||R - RS||_F^2 + \frac{\beta}{2}||S||_F^2 + \lambda||S||_1 \\ \text{s.t.} \quad & S \geq 0 \\ & diag(S) = 0 \end{aligned}$$

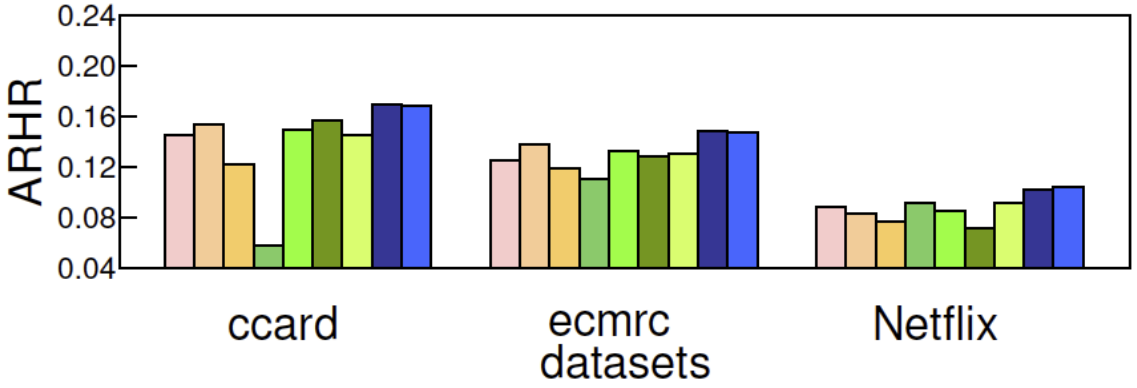
  - Compute rating as:

$$\hat{r}_{ui} = \sum_{j \in R_u} r_{uj} s_{ji}$$

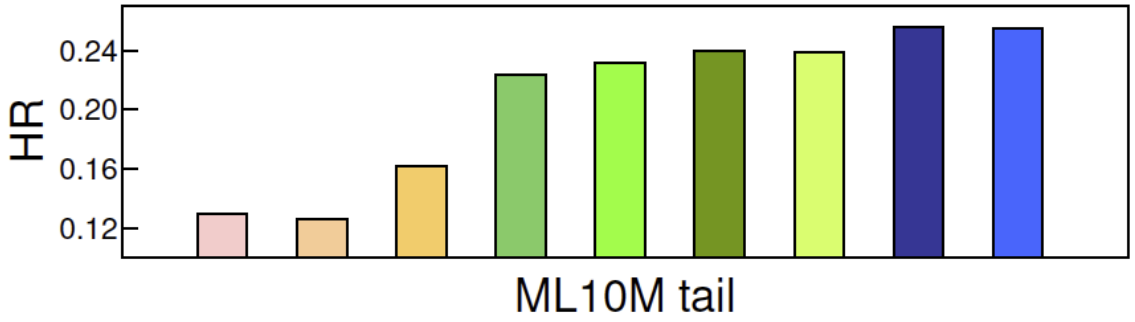
SLIM - Performance

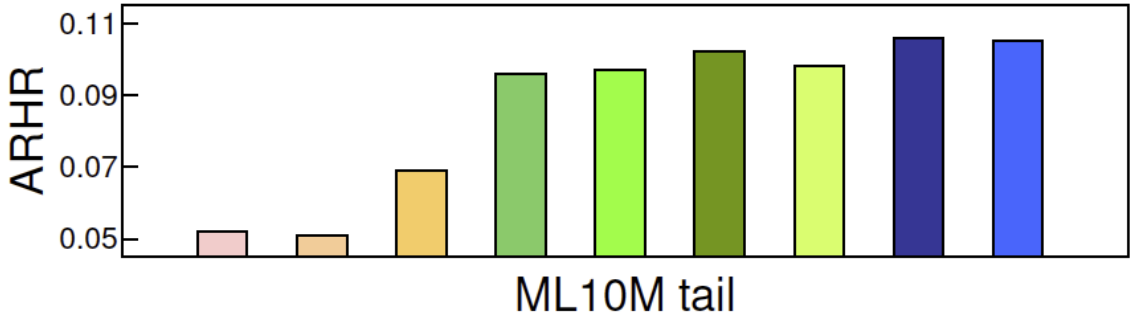
Overall Performance





Long Tail Performance





itemkNN
itemprob
userkNN
PureSVD
WRMF
BPRMF
BPRkNN
SLIM
fsSLIM

SLIM – Sparse Linear Model

- The SLIM model can be computed efficiently
  - Each column of S can be computed independently.
  - The solution of a column can “warm start” other similar columns.
  - Item neighbors can be used for initial “feature” selection.
- Slim extensions:
  - Low-rank constraint on S as an alternative way to control model complexity (FISM).
    - Via the product of two low-rank matrices or minimizing the nuclear norm of S.
  - Incorporate item side-information into the model.
  - Incorporate temporal information.
  - Higher-order regression (HOSLIM).
  - Fusion of global and local SLIM models (GL-SLIM).

Explanations in recommender systems

Motivation

- "The digital camera *Profishot* is a must-buy for you because . . . ."
- Why should recommender systems deal with explanations at all?
- The answer is related to the two parties providing and receiving recommendations:
  - A selling agent may be interested in promoting particular products
  - A buying agent is concerned about making the right buying decision

Scaling up Recommender Systems

David C. Anastasiu, San Jose State University [david.anastasiu@sjsu.edu]

Summarizing recent trends

- Boundaries between neighbor and latent factor models have become blurred.
- Both classes of methods have borrowed ideas from the other, and
- New methods have been developed that combine them

- Neighbor methods may compute similarities in the latest space

$$R \sim PQ', \ \mathrm{sim}(i,j) = q_i q_j', \ \mathrm{sim}(u,v) = p_u p_v'$$

- Factorization methods rely on similar items to derive a latent representation of the user

$$\hat{r}_u i = \mu + b_u + b_i + \left(\frac{1}{|U|^\alpha} \sum_{j \in U} r_{ui} qj \right) q_i'$$

Summarizing recent trends

- Recommendation is concerned with learning from noisy observations (x, y), where f(x) has to be determined such that the error is minimal.

$$f(x) = \hat{y} \qquad \sum_{\hat{y}} (\hat{y} - y)^2$$

- A variety of different learning strategies have been applied trying to estimate f(x)
  - Non parametric neighborhood models
  - MF models, SVMs, Neural Networks, Bayesian Networks,…

Current State-of-the-Art

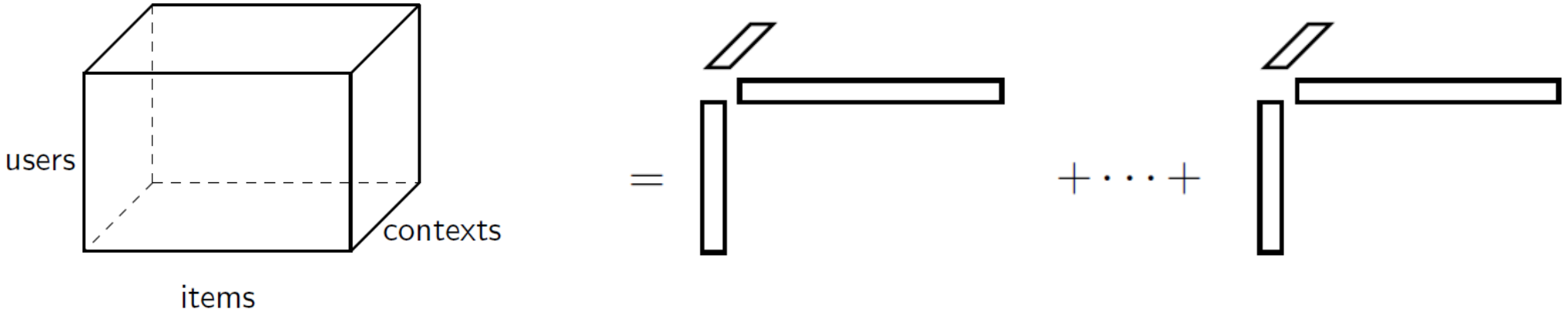
- Rating Prediction
  - Factorization-based approaches are highly effective:
    - Good prediction performance, efficient training, can incorporate side-information and diverse objectives, etc.
  - Tensor factorization approaches are being heavily researched.
- Top-N Recommendation
  - No clear winning strategy has emerged.
    - Old item-based nearest neighbor methods still outperform significantly more sophisticated methods
  - There is significant ongoing research on ranking loss functions.

How Can We Scale Up?

- Sampling
  - Estimate only item factors from a subset of users & compute user factors on the fly.
  - Warm-start the search over the regularization space
    - For MF, use SVD to eliminate bad local minima
    - For SLIM, use previous solutions and/or solutions for similar items.
  - Sample the users with a goal of getting reliable item-based models
    - Either item-factors or item-item models.
- Parallelism
  - How do we effectively split work between processes?
  - Can we avoid work conflicts?
- Approximation
  - Find approximate neighbors and/or
  - Approximate the computation of the proximity function

Tensor Factorization

- Tensors are like matrices but have >2 dimensions.
- We can model ratings as a $user \times item \times context$ tensor.



- Canonical Polyadic Decomposition (CPD)
  - Given an M-way tensor R
  - Compute the low-rank matrices $P_1 \ldots P_M$
  - Whose product re-produces R. Elementwise, $r_{i_1 \ldots i_M} = \sum_{k=1}^{K} \prod_{m=1}^{M} p_{i_m k}^{(m)}$
  - The CPD is essentially unique

Efficient CPD computation

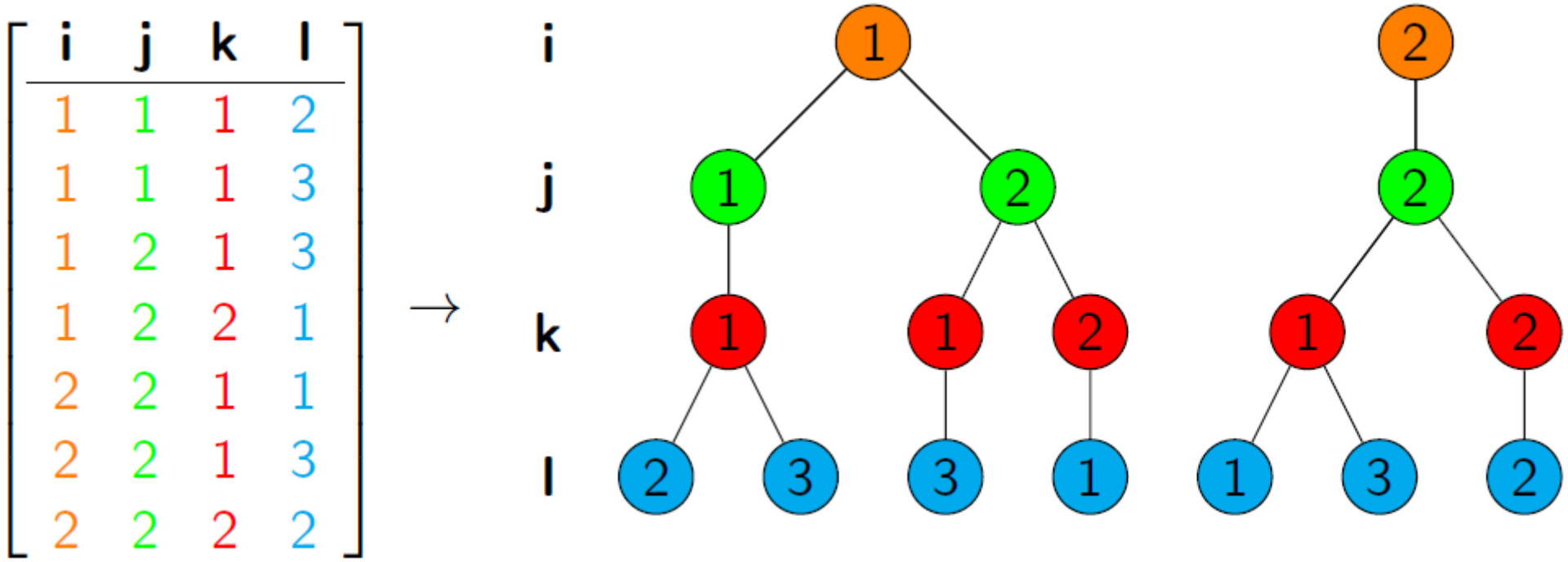
- Alternating Least Squares method similar to the one for MF
- The bottleneck in the TF case is the Matricized Tensor Times Katri-Rao Product (MTTKRP)

$$
\begin{aligned}
\mathbf{W} &\leftarrow (\mathbf{P}^{(M)} \odot \cdots \odot \mathbf{P}^{(2)}) \\
\mathbf{P}^{(1)} &= \underset{\mathbf{P}^{(1)}}{\operatorname{argmin}} \, ||\mathbf{R}_{(1)} - \mathbf{P}^{(1)}\mathbf{W}^T||_F^2 + \lambda_1 ||\mathbf{P}^{(1)}||_F^2 \\
&= \underbrace{\mathbf{R}_{(1)}\mathbf{W}}_{\text{MTTKRP}} \underbrace{\left(\mathbf{W}^T\mathbf{W} + \lambda_1 \mathbf{I}\right)^{-1}}_{\text{Normal EQs } (K \times K)}
\end{aligned}
$$

- Explicitly forming $(\mathbf{P}^{(M)} \odot \cdots \odot \mathbf{P}^{(2)})$ is infeasible, so it's done in-place.

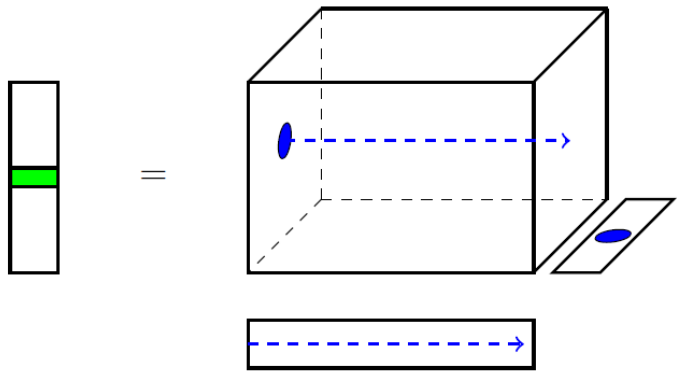
Sparse Tensor Representation

- Smith'15 – Compressed Sparse Fiber (CSF)
  - Hierarchical storage format for tensors (extension of the CSR concept)
  - Paths from roots to leaves encode non-zero coordinates

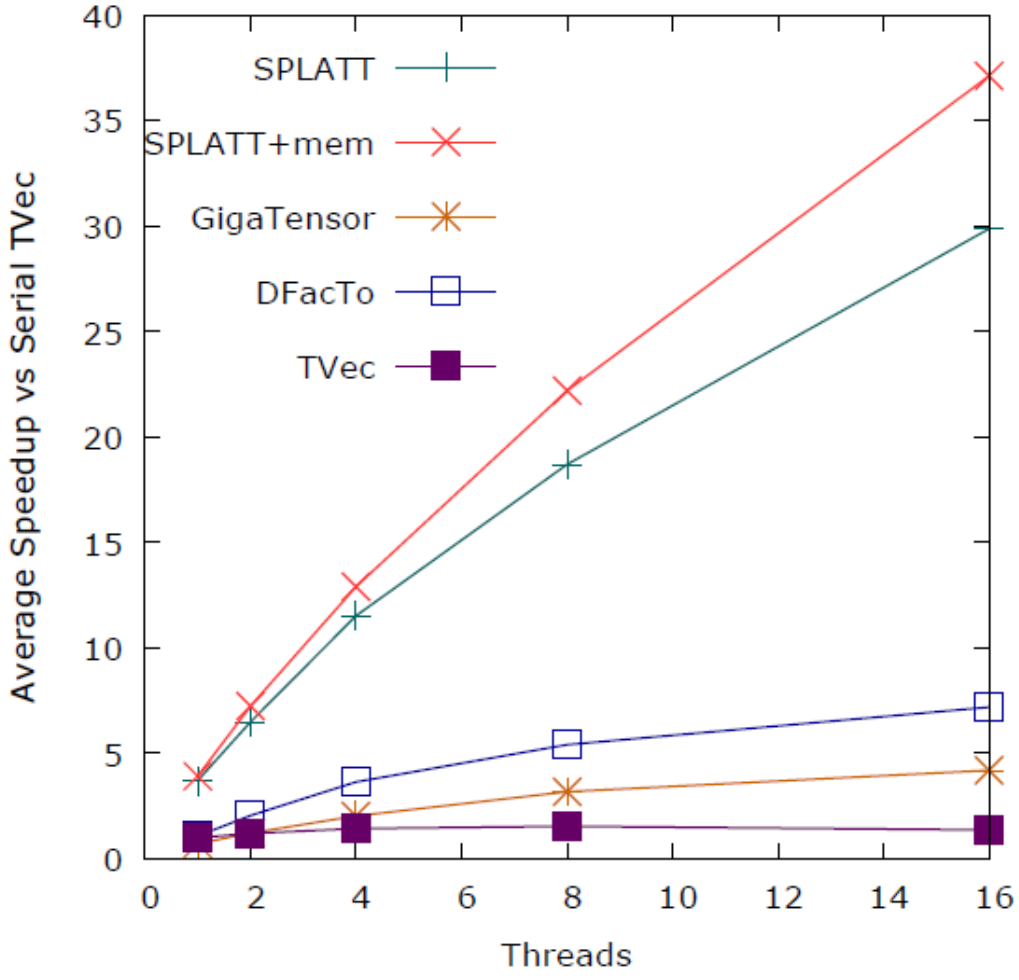


Parallel MTTKRP

- Smith'15 – The Surprisingly ParalleL spArse Tensor Toolkit (SPLATT)
  - Uses the CSF data structure to reduce operation count during MTTKRP
  - SPLATT processes fibers instead of non-zeros



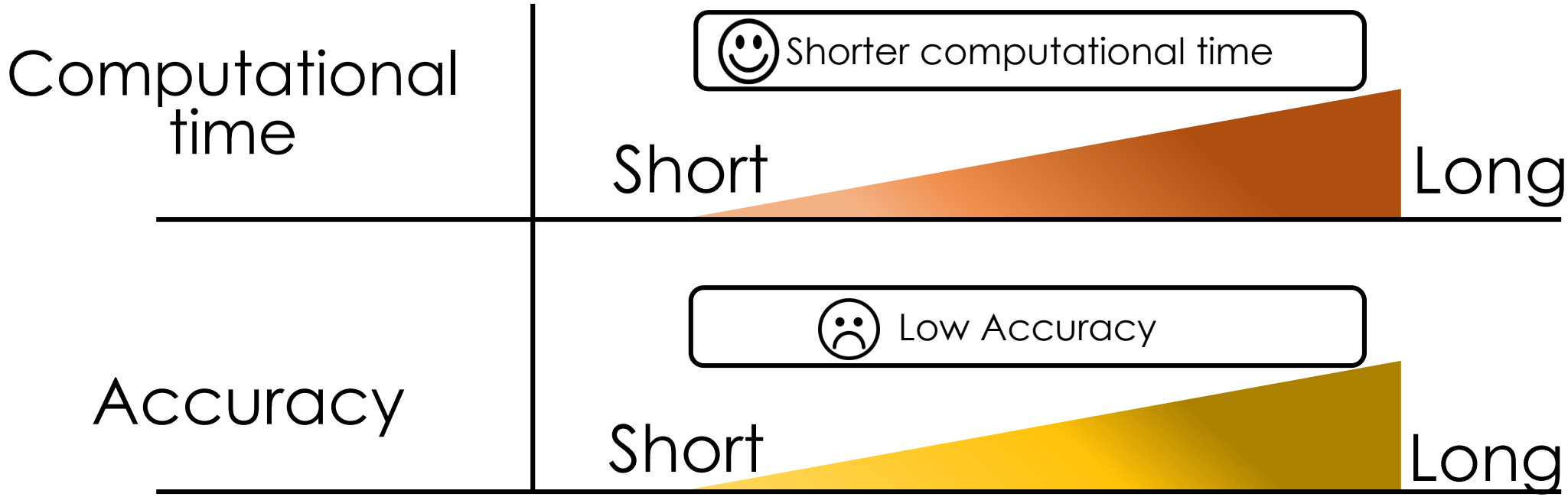
  - Techniques such as reordering and cache tiling improve performance



Sparsity in Tensors

- Tensors are often extremely sparse
- CSF and SPLATT techniques have been applied to many other problems, including
  - Large-scale distributed Alternating Least Squares
  - Constrained tensor factorization
  - Accelerating the Tucker decomposition
  - Streaming tensor factorization

Effect of nearest neighbor search approximation on quality of recommendation



Trade-off between efficiency and effectiveness

Is there any evidence that approximate methods have an advantage over exact methods?

Are approximate methods "good enough" as they claim to be? Does it solve real world nearest neighbor problems efficiently?

Research Question

In using Approximate nearest neighbor search as part of a recommendation pipeline,

what is the trade-off between the search performance and the effectiveness of the recommendation engine?

Problem setting

ItemKNN

- Collaborative Filtering
- Cosine Similarity

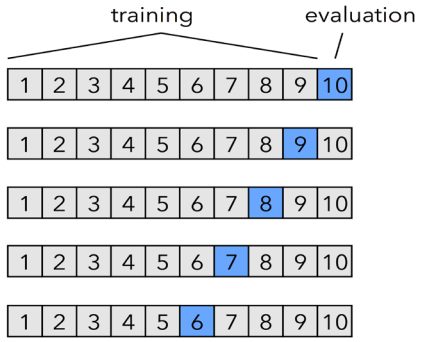
Nearest Neighbor Search

L2KNNG - Exact Cosine KNNG Construction

L2KNNGApprox – Heuristic Approximate KNNG Construction

Evaluation Methodology

5-fold Leave One Out Cross Validation (LOOCV)



Evaluation metric for Recommendation

Hit rate (HR) = #hits / #users

#hits - the number of users whose item in the testing set is recommended
#users - the total number of users

Evaluation metric for nearest neighbor search

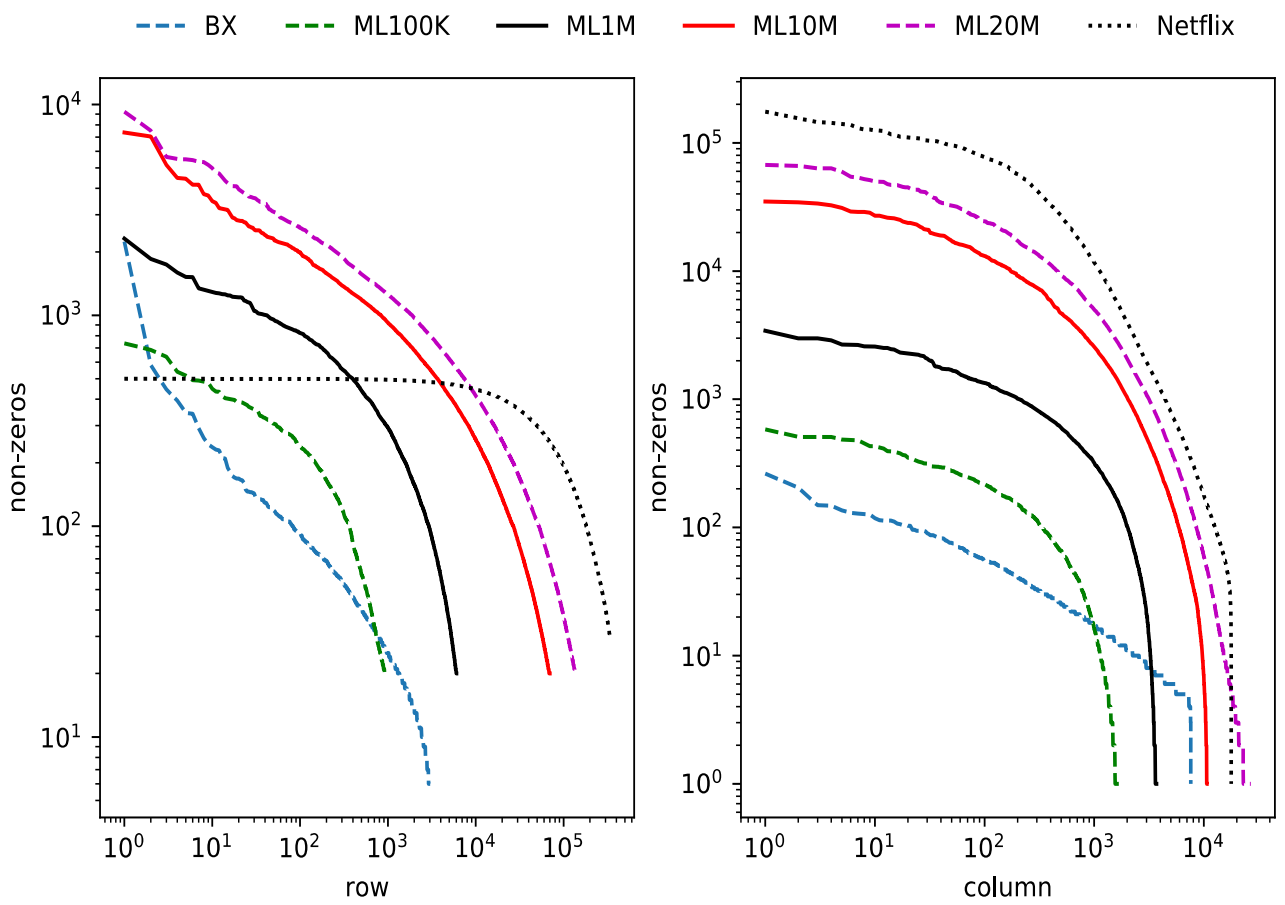
Approximation Quality – Recall
i.e., Ratio of exact *k*-nearest neighbors in the current *k* nearest neighbor approximation.

Recall for exact – 1.0

Dataset Characteristics

Dataset	Users	Items	Transactions
BX	3,586	7,602	84,981
ML100K	943	1,682	100,000
ML1M	6,040	3, 706	1,000,209
ML10M	69,878	10,677	10,000,054
ML20M	138,493	26,744	20,000,263
Netflix	336,914	17,770	51,937,015

Transfer into Implicit ratings



Baseline Methods

MF - Matrix Factorization.

factorizes a large matrix into two small matrices called features

BPR - Bayesian Personalized ranking.

provides user with a ranked list of items.

WARP - Weighted Approximate-Rank Pairwise loss

K-OS - K-Order Statistic (K-OS) loss

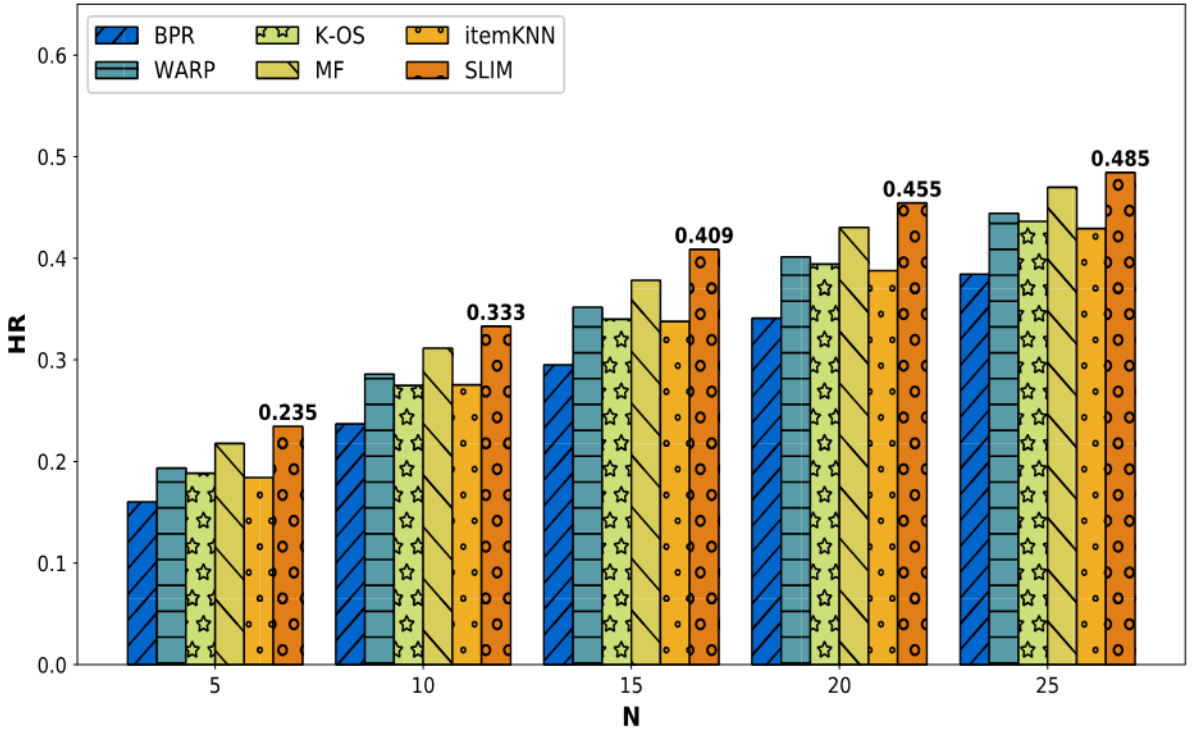
SLIM - Sparse LInear Method

ITEMKNN - Item based neighborhood collaborative filtering algorithm

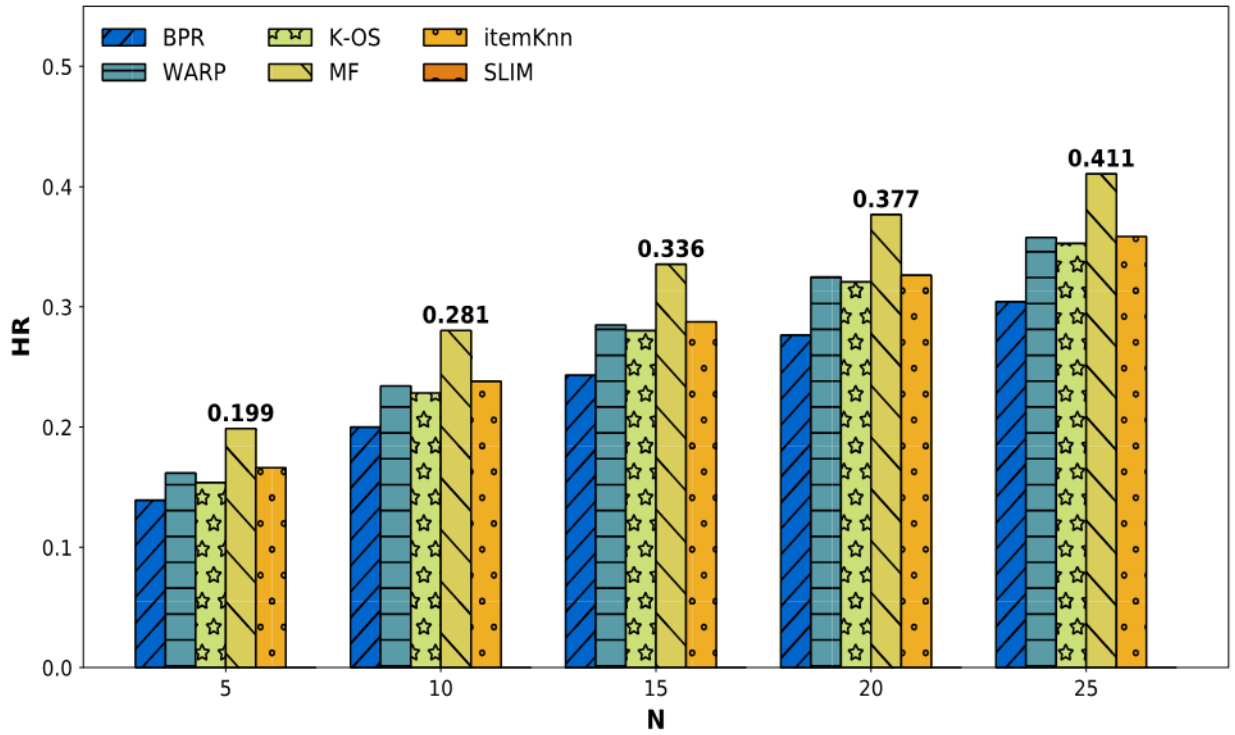
Experimental Results - 1

Top-N Recommender Performance

ML100K



ML10M



ItemKNN vs. Best Performing baseline

Dataset	N = 5	N = 10	N = 15	N = 20	N = 25	P-value
BX	0.008	0.009	0.014	0.016	0.019	**0.0032**
ML100K	0.050	0.057	0.070	0.066	0.055	< 0.0001
ML1M	0.048	0.063	0.069	0.070	0.073	< 0.0001
ML10M	0.032	0.042	0.047	0.050	0.052	**0.0002**
ML20M	0.033	0.040	0.045	0.047	0.048	**0.0001**
Netflix	0.0	0.0	0.0	0.0	0.0	-

Performance Comparison:

BestHR – ItemKNN's HR

Paired t -test

Null hypothesis – the average of the performance difference between the itemKNN and the best performing baseline is zero

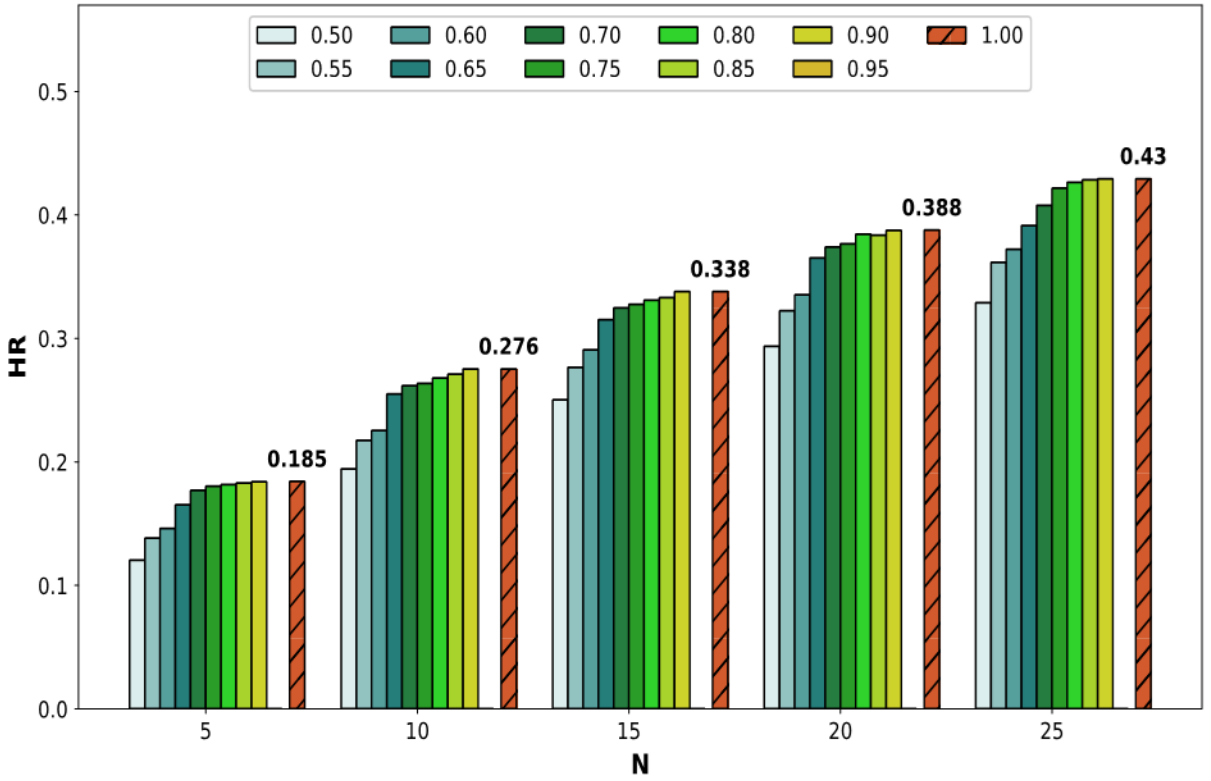
p-value >= 0.0001 – fail to reject hypothesis

p-value < 0.0001 – rejects hypothesis

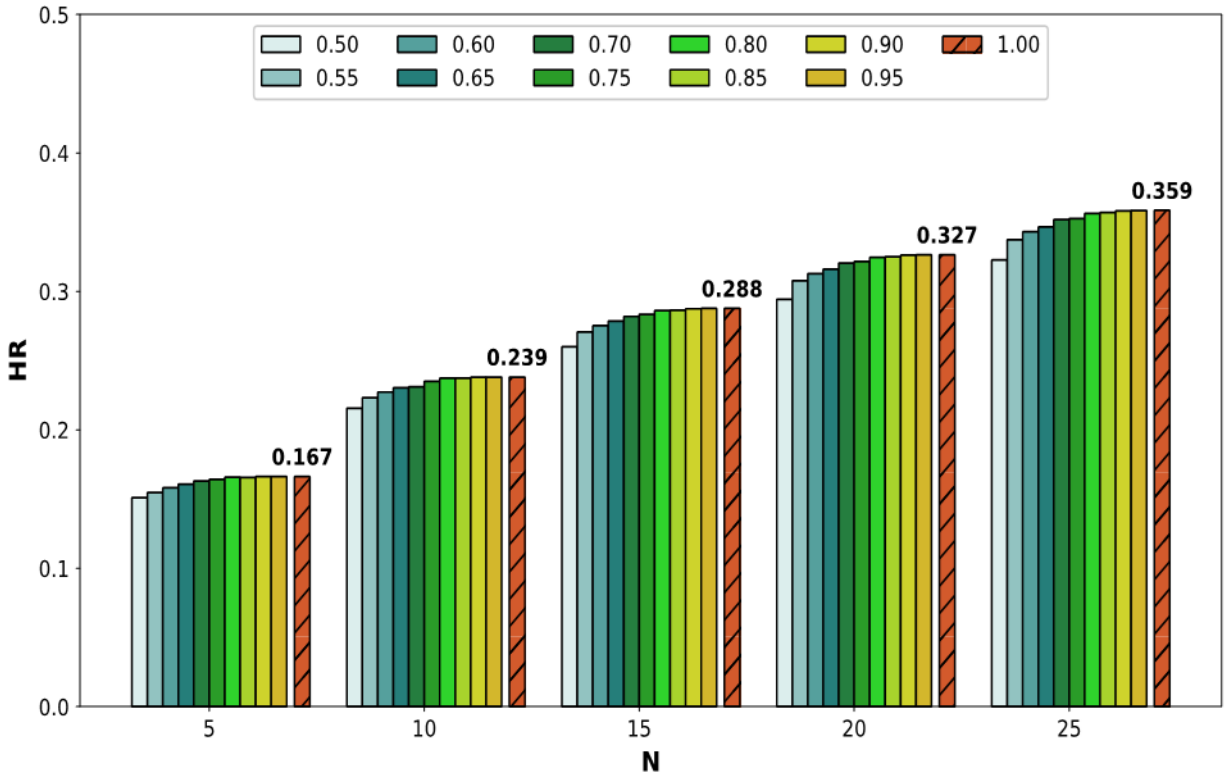
Experimental Results - 2

ItemKNN Recommender Performance

ML100K



ML10M

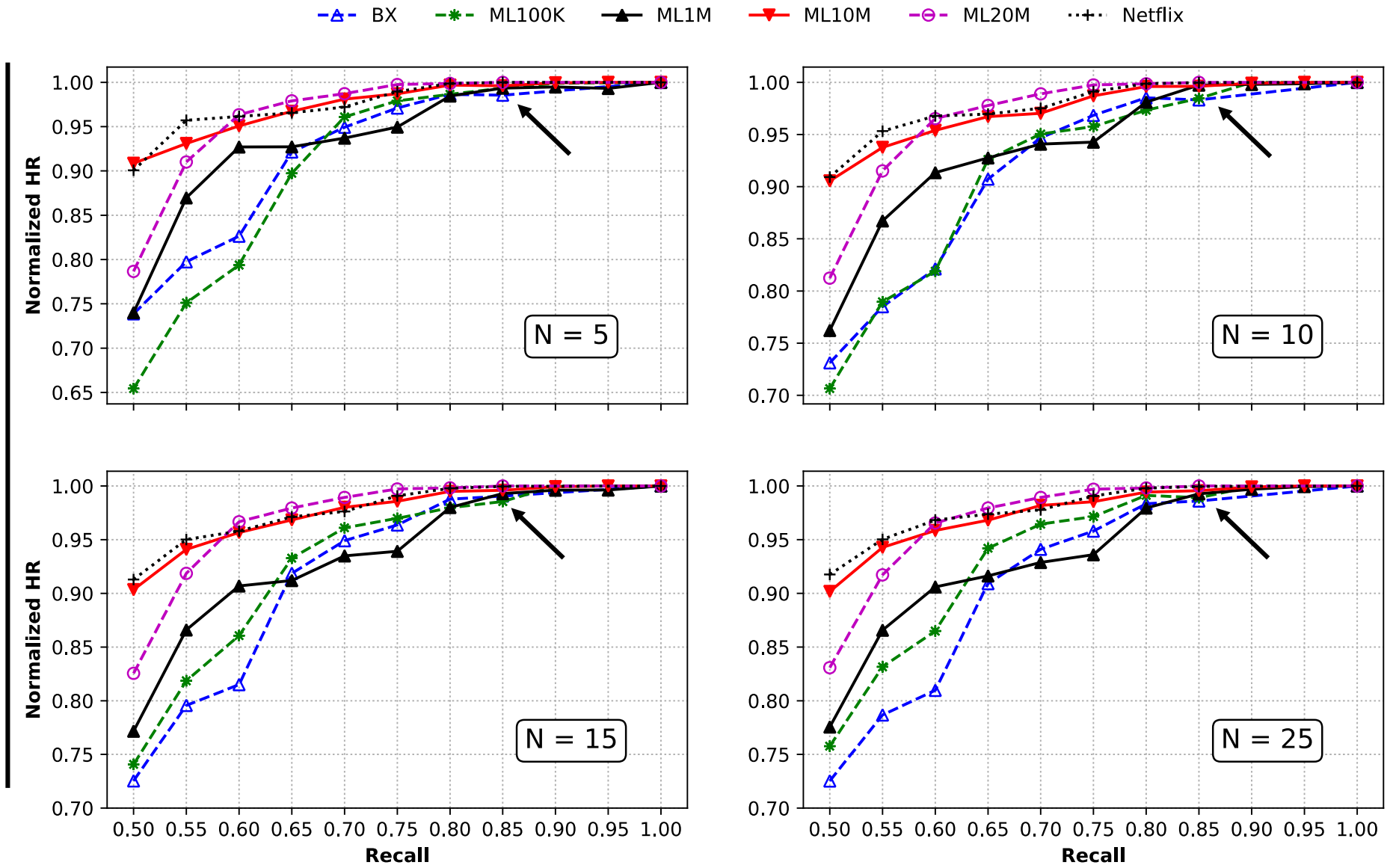


Experimental Results - 3

Recall

vs.

Normalized HR

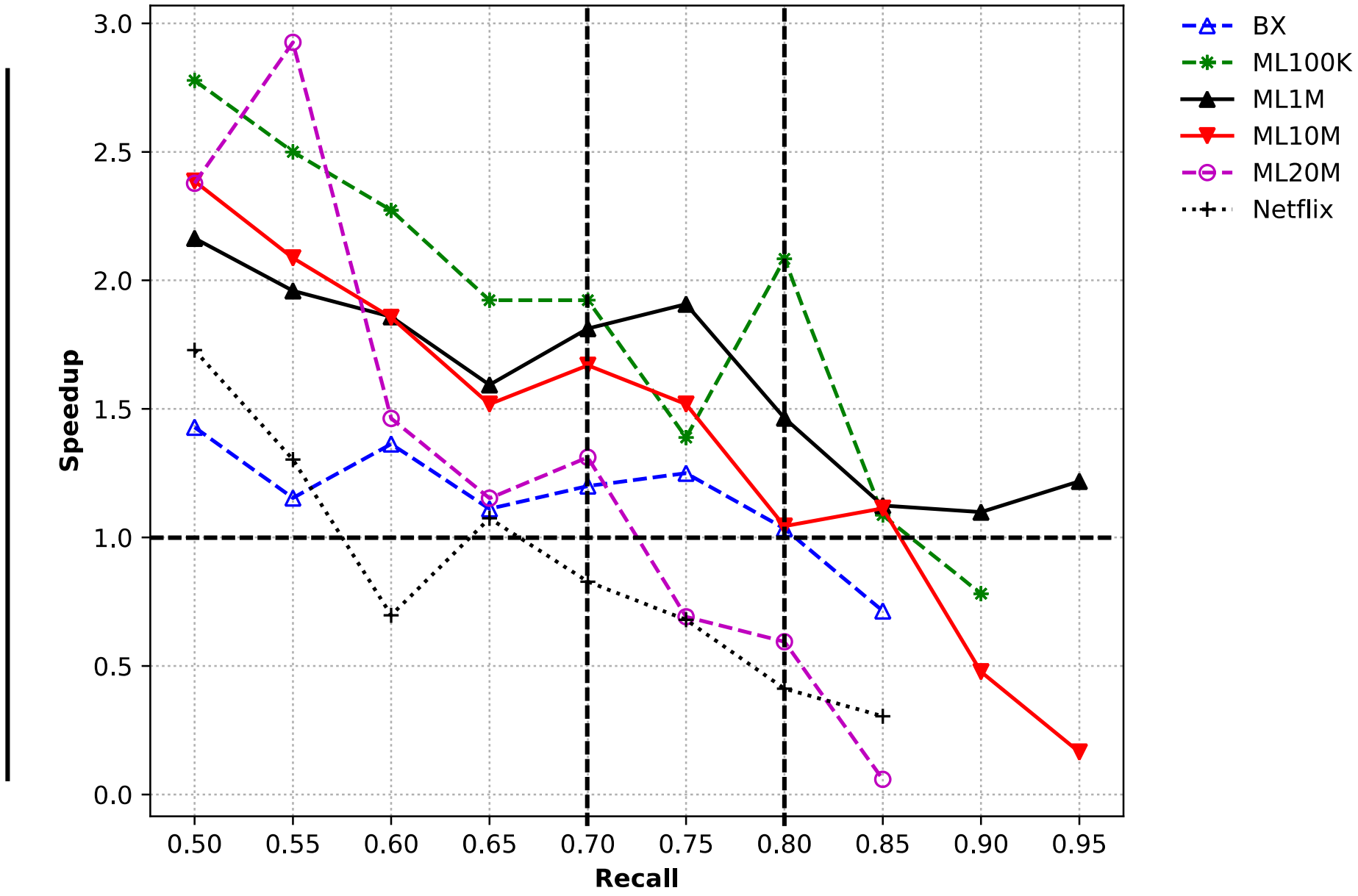


Recall has to be at least 85% (Normalized HR – 0.97)

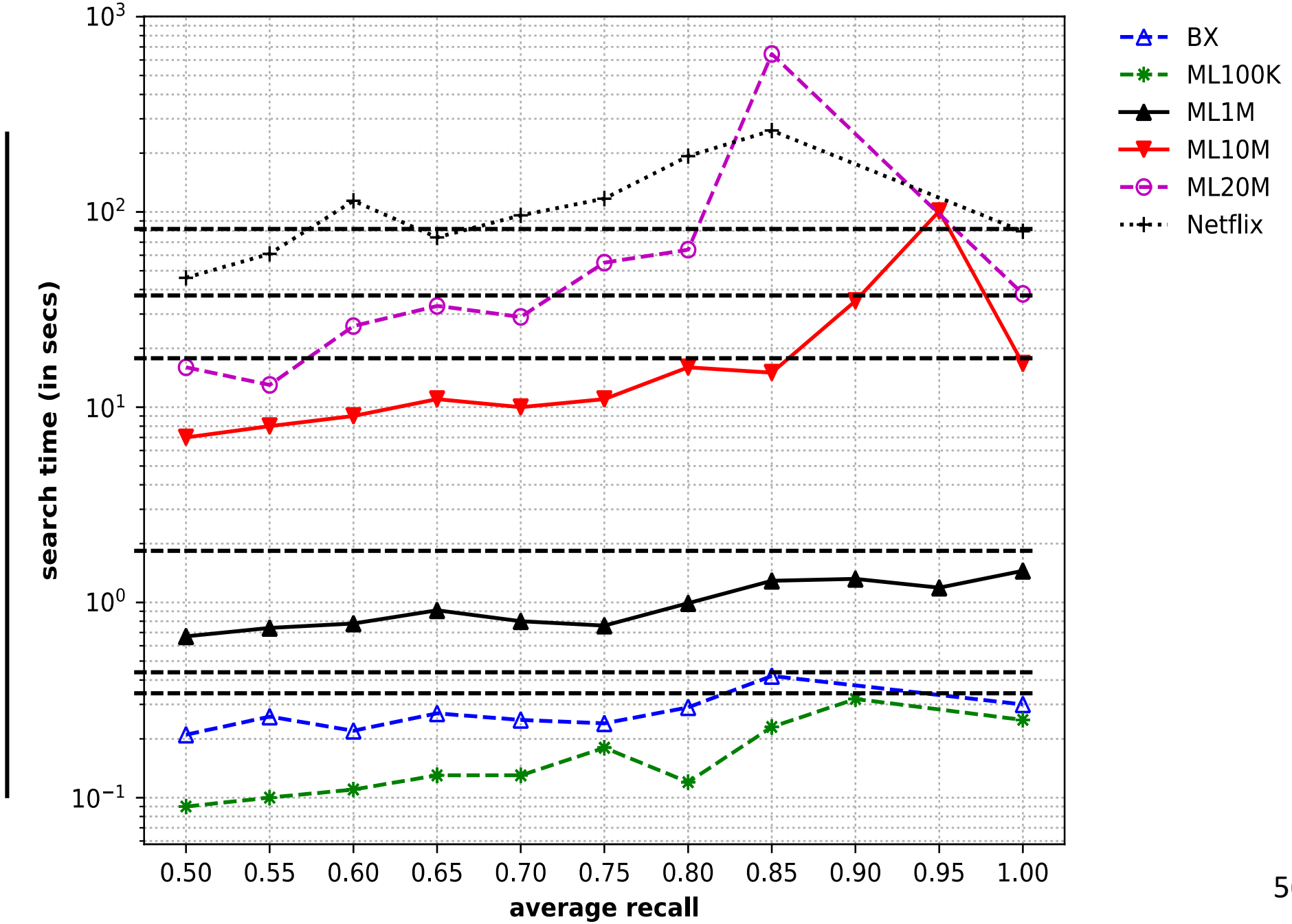
Recall

vs.

Speedup



Effectiveness
vs.
Efficiency



Findings

High recall levels (above 80%) - Search time of the approximate may be > exact method

Recall levels between 70% and 80% - approximate methods are “good enough”

Recall levels below 70% - Significantly worse recommendation performance

Not all the datasets achieve near-perfect recall using approximate methods.

Computation cost to reach the desired recall for an approximate method may be greater than exact search using efficient searcher

Acknowledgements

- Students:

Saranya Rajan, MS

Alex Richards, BS

References

[1] D. C. Anastasiu and G. Karypis, “L2knng: Fast exact k-nearest neighbor graph construction with l2-norm pruning,” in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM ’15, (New York, NY, USA), pp. 791–800, ACM, 2015.

[2] X. Ning and G. Karypis, “Slim: Sparse linear methods for top-n recommender systems,” in Proceedings of the 2011 IEEE 11th International Conference on Data Mining, ICDM ’11, (Washington, DC, USA), pp. 497–506, IEEE Computer Society, 2011.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in Proceedings of the 10th International Conference on World Wide Web, WWW ’01, (New York, NY, USA), pp. 285–295, ACM, 2001.

[4] Shaden Smith, Niranjay Ravindran, Nicholas D Sidiropoulos, and George Karypis. 2015. SPLATT: Efficient and parallel sparse tensor-matrix multiplication. In 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS'15).