Deep Graph Embeddings in Recommender Systems

Thesis Defense

Data-Driven Decision-Making Lab

Soon Chee Loong

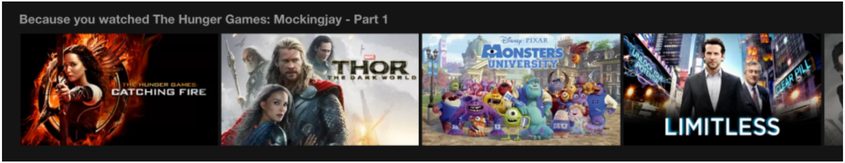
2019, August 26

University of Toronto
Supervisor: Professor Scott Sanner
Defense Committee: Professor Scott Sanner, Professor Chi-Guhn Lee, Professor Timothy Chan

Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

Recommender Systems for the World

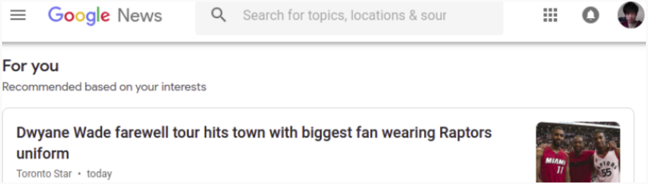
Recommender Systems in Industry



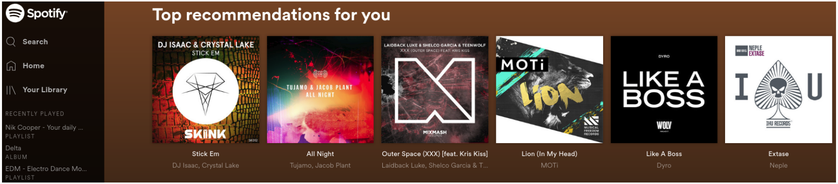
Netflix Movie Recommender worth about $1 billion.

Amazon Shopping Recommender accounts for 35% of sales.

Recommender Systems in Industry



Google News Recommender generates 38% more clicks.

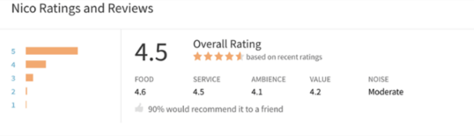


Spotify Music Recommender streams 5 billion tracks in a year.

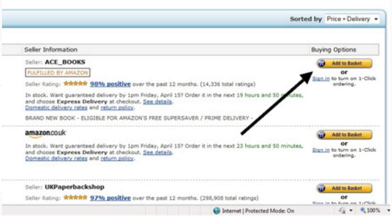
Recommendation Datasets and Challenges

Recommendation Datasets and Challenges

What type of data does recommender systems receive?



Ratings



Purchases

- Explicit Data: ratings, right swipes, likes
- Implicit Data: clicks, views, purchases

Challenge: **One-class feedback** for implicit data.

- 1 represents positive feedback
- 0 represents negative or unaware feedback

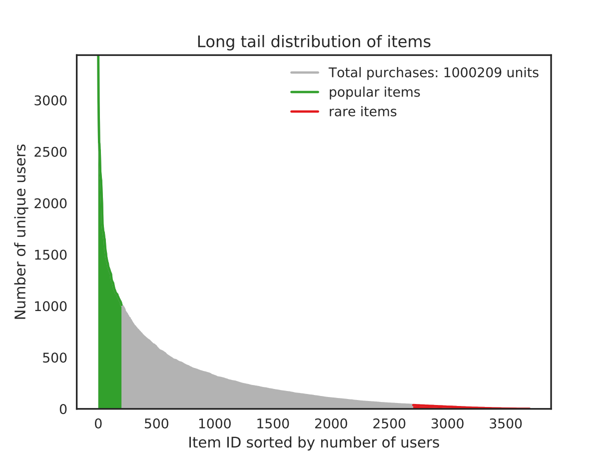
Recommendation Datasets and Challenges

Dataset	$\lvert U \rvert$	$\lvert I \rvert$	$\lvert R \rvert$	$\frac{\lvert R \rvert}{\lvert U \rvert * \lvert I \rvert}$
MovieLens-100k	943	1682	100000	6.30×10^{-2}
MovieLens-1m	6040	3706	1000209	4.46×10^{-3}
BookCrossing	7721	5000	253967	6.58×10^{-3}
Amazon Video Games	7926	5000	107359	2.71×10^{-3}

Each user interacts with only a few items.

Challenge: **Scalability** due to huge number of users and items.
Challenge: **Sparsity** due to relatively low number of interactions.

Recommendation Datasets and Challenges



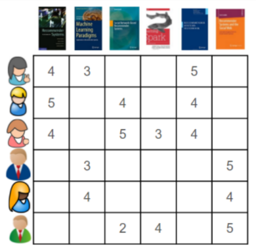
A niche set of popular items dominate the dataset

Challenge: **Imbalanced** item distribution.

Recommendation Problem

Recommendation Problem Formulation

$R =$ (sparse) Rating Matrix
$U =$ Set of users
$I =$ Set of items
$R_{train} =$ Train Rating Matrix
$R_{test} =$ Test Rating Matrix
$R = R_{train} \cup R_{test}$, $R_{train} \cap R_{test} = \emptyset$
$\hat{R} = f(R_{train}) =$ Predicted Matrix
$\hat{R}$ is dense.
$f =$ Recommender System

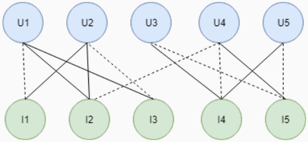


User-Item Rating Matrix

Predict scores for missing entries of the Rating Matrix, $R_{missing} \approx \hat{R}$.
Then, recommend the Top-K missing entries for each user.

Recommendation Problem Formulation

G = Bipartite Graph
V^U = Set of users nodes
V^I = Set of items nodes
E = (sparse) interaction edges
$E = E_{train} \cup E_{test}$, $E_{train} \cap E_{test} = \emptyset$
$\hat{E_{test}} = f(E_{train})$ = Link Prediction
$\hat{E_{test}}$ is dense.
f = Recommender System

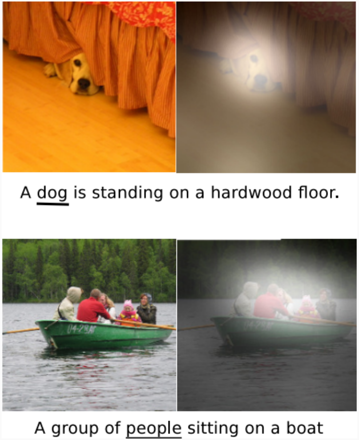


User-Item Bipartite Graph

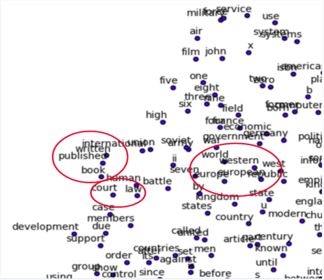
Predict scores for missing edges in the bipartite graph.
Then, recommend the Top-K missing edges for each user nodes.

Deep Graph Embeddings

Success of Deep Embeddings



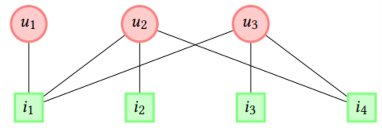
CNN Embeddings



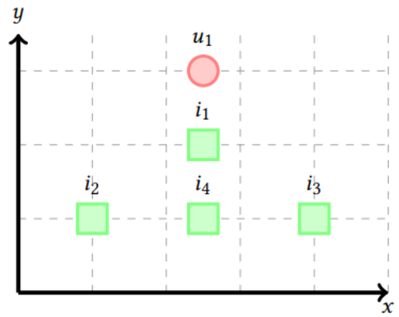
SGNS embeddings

Generalize Deep Embeddings to Deep Graph Embeddings

Spectral Graph Convolution



Bipartite Graph

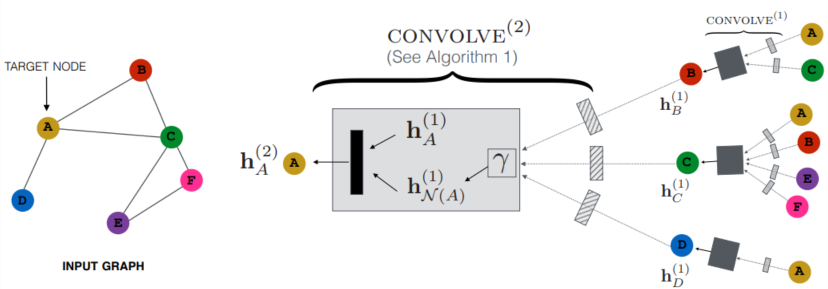


Spectral Convolution Embeddings

Spectral domain carries connectivity information.
u_1 is more connected to i_4 compared to i_2 or i_3.

Generalize Deep Embeddings to Deep Graph Embeddings

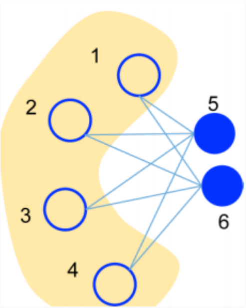
Spatial Graph Convolution



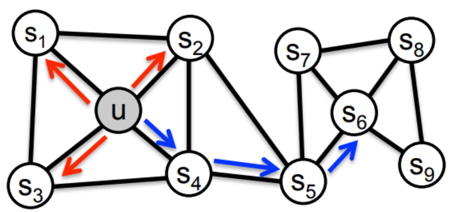
Node's embedding is a convolution of its neighbours convolved embeddings.

Generalize Deep Embeddings to Deep Graph Embeddings

DeepWalk



Random Walk on nodes



Breadth-First vs Depth-First Walks

Nodes on random walk are pulled closer in embedding space.

Research Questions

Research Questions

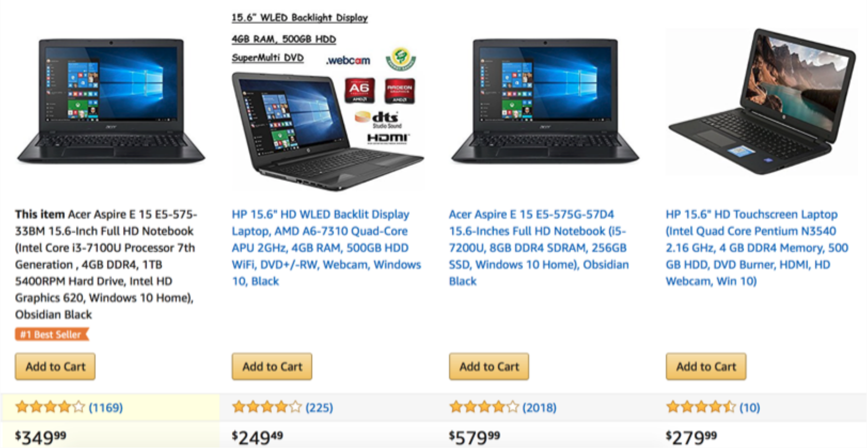
- What are the properties of graph embeddings?
- How do graph embeddings compare to state-of-the-art algorithms?
- How to extend graph embeddings for personalized recommendations?

Analyzing Embeddings of Recommender Systems

Applications of Embeddings

Debugging: Embeddings within a category should be clustered.

Applications of Embeddings

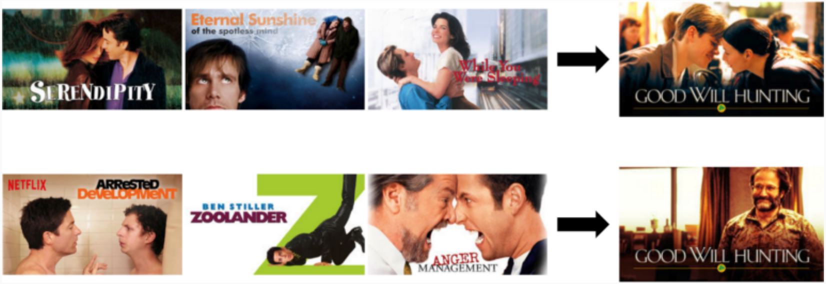


Item Retrieval: Retrieving item substitutes for comparison before purchase.

Applications of Embeddings

Interpretability: Explain recommendations based on similarity to item.

Applications of Embeddings



Personalization: Similar to average a user's past interactions.

Item Similarity Matrix

Item Similarity, S^I

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

$$S^I = W^I \cdot (W^I)^T$$

$$W^I \in R^{|I| \times d}$$

S^I must respect metric spaces.

- Identity Similarity
- Clustering Coherency
- Similarity Propagation
- Average Similarity

Unsupervised Similarity Analysis

Table 5.1: Identity Similarity Analysis

Model	IdentitySimilarity
DeepWalk	**0.9969493593654668**
ProfitWalk	0.9920683343502136
KNN	0.9286150091519219
WRMF	0.8425869432580841
BPR	0.7809640024405126
GC-MC	0.7559487492373398
PMF	0.6168395363026236
PureSVD	0.5247101891397193
DeepRec	0.07931665649786455
SpectralCF	0.0018303843807199512
Popular	0.0006101281269066504

Table 5.2: MovieLens-100k Identity Similarity

Model	IdentitySimilarity
KNN	**0.9771677086164718**
DeepWalk	0.9747213916825225
ProfitWalk	0.9567817341668932
WRMF	0.9141070943191084
PMF	0.8486001630877956
GC-MC	0.39657515629247075
PureSVD	0.2304974177765697
BPR	0.0008154389779831476
DeepRec	0.0005436259853220984
SpectralCF	0.0005436259853220984
Popular	0.0002718129926610492

Table 5.3: MovieLens-1m Identity Similarity

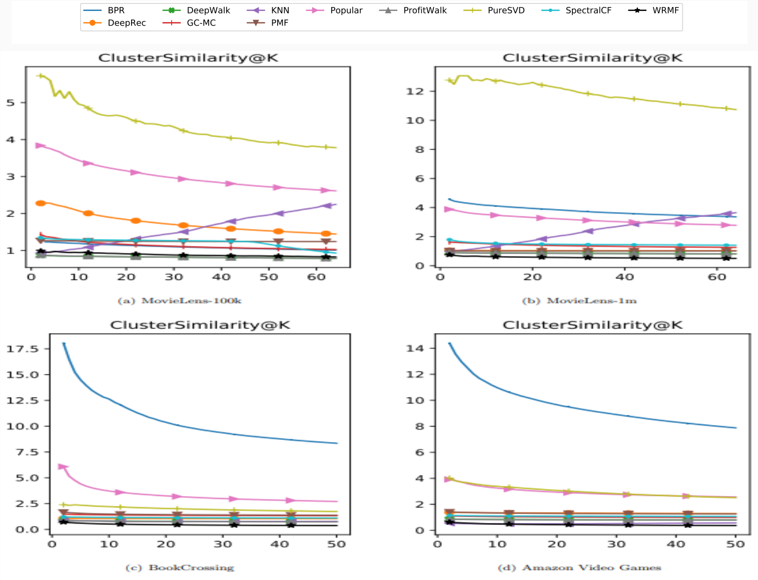
Model	IdentitySimilarity
DeepWalk	**0.9936**
KNN	0.984
ProfitWalk	0.9338
WRMF	0.8808
PMF	0.8258
GC-MC	0.7444
PureSVD	0.181
DeepRec	0.0502
BPR	0.0008
SpectralCF	0.0002
Popular	0.0002

Table 5.4: BookCrossing Identity Similarity

Model	IdentitySimilarity
DeepWalk	**0.9959895728895127**
KNN	0.9921796671345499
ProfitWalk	0.9797473430920393
WRMF	0.9422498496089834
GC-MC	0.9049528774814518
PMF	0.5660717866452777
DeepRec	0.23180268698616402
PureSVD	0.09123721676358532
BPR	0.0010026067776218166
SpectralCF	0.00020052135552436334
Popular	0.00020052135552436334

Table 5.5: Amazon Video Games Identity Sim.

Unsupervised Similarity Analysis



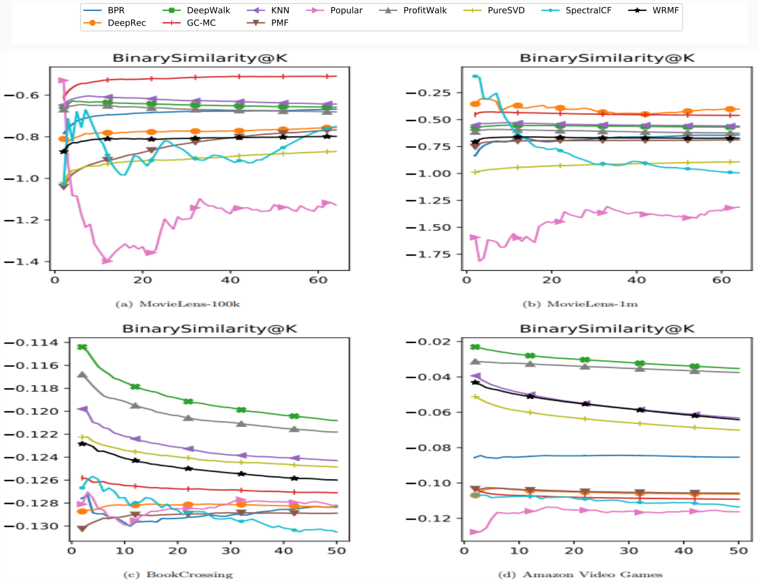
(a) MovieLens-100k

(b) MovieLens-1m

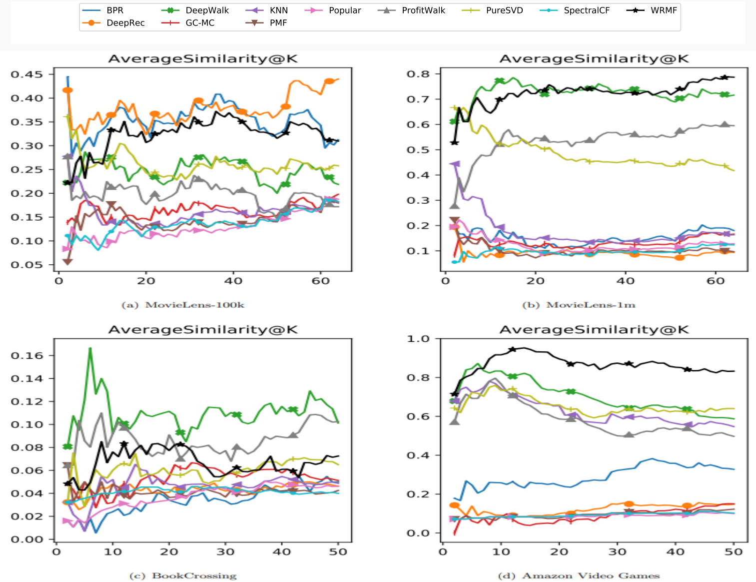
(c) BookCrossing

(d) Amazon Video Games

Supervised Similarity Analysis



Supervised Similarity Analysis



(a) MovieLens-100k

(b) MovieLens-1m

(c) BookCrossing

(d) Amazon Video Games

Top-K Ranking Recommendation

Top-K Ranking Recommendation

Model	R-Precision
BPR	**0.22579149237442533**
WRMF	0.1841242282690616
DeepRec	0.17368522306349038
ProfitWalk	0.14920530109228317
PureSVD	0.134121742616618
DeepWalk	0.13407648848231915
Popular	0.12942480308774157
PMF	0.08037262846870519
SpectralCF	0.04243381758161209
GC-MC	0.04206175310215691
KNN	0.028821922164207498

Table 6.3: MovieLens-100k R-Precision

Model	R-Precision
PureSVD	**0.16712794423094776**
ProfitWalk	0.14874195923762576
DeepWalk	0.1376663334186439
WRMF	0.13525588818481504
BPR	0.125815116748618
Popular	0.10634943136289311
DeepRec	0.07818605888941238
SpectralCF	0.07615882526088245
KNN	0.053240445295168304
PMF	0.04351476384548103
GC-MC	0.016550159020925787

Table 6.4: MovieLens-1m R-Precision

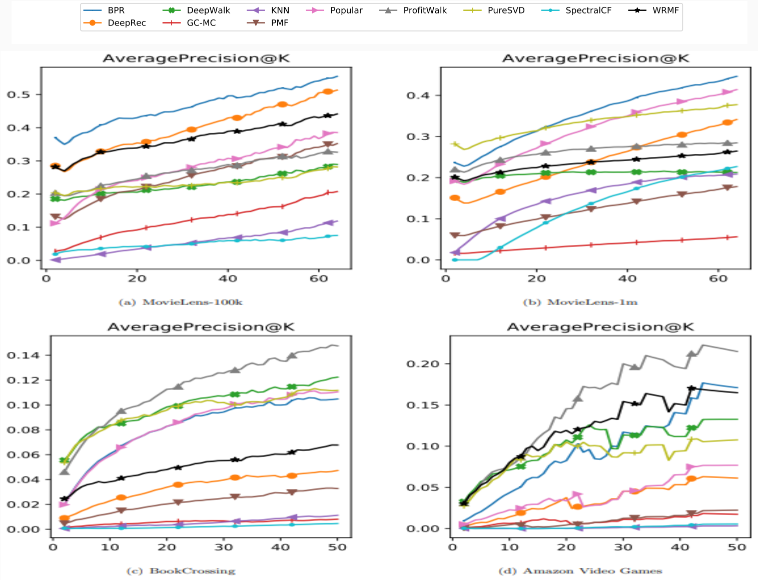
Model	R-Precision
DeepWalk	**0.03537717620367315**
ProfitWalk	0.028105778250450776
PureSVD	0.027979193552098422
BPR	0.013798620384167482
Popular	0.013788971609372492
WRMF	0.01528680219247615
DeepRec	0.004230860304362016
PMF	0.0029544114180469483
SpectralCF	0.0013064369976799556
GC-MC	0.0011663642887627282
KNN	0.0007282023657656779

Table 6.5: BookCrossing R-Precision

Model	R-Precision
DeepWalk	**0.027476066515407804**
ProfitWalk	0.024723589142794496
WRMF	0.02267812923406185
PureSVD	0.01942766456071679
BPR	0.006751490554507684
DeepRec	0.0030388102916909794
Popular	0.00293726167961445
SpectralCF	0.001033168189966927
GC-MC	0.0007350850942178361
PMF	0.0005112025682056364
KNN	0.0001622652973791078

Table 6.6: Amazon Video Games R-Precision

Top-K Ranking Recommendation



(a) MovieLens-100k

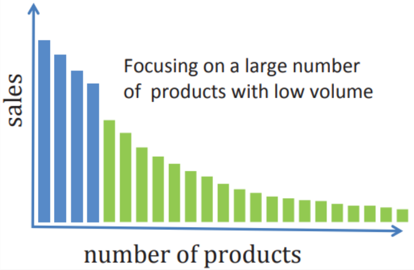
(b) MovieLens-1m

(c) BookCrossing

(d) Amazon Video Games

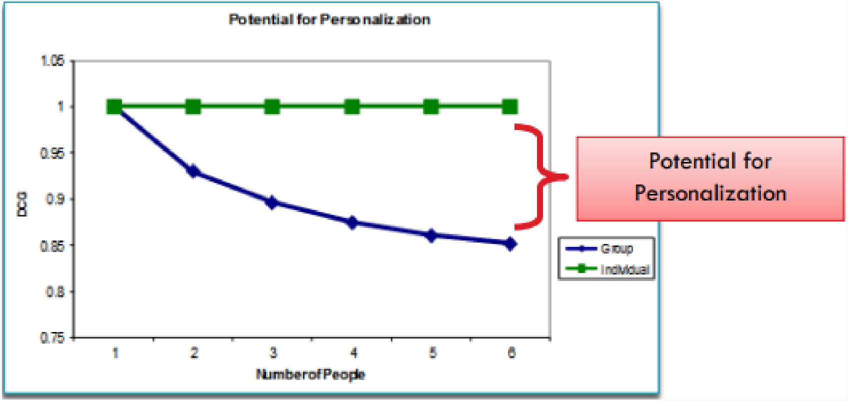
Profitability in Long-Tail Recommendations

Profitability in Long-Tail Recommendations



Long Tail [1]

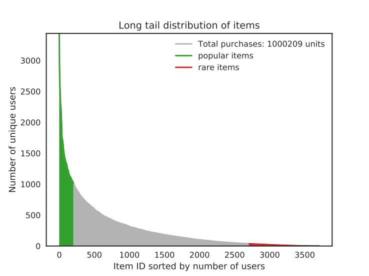
Personalization



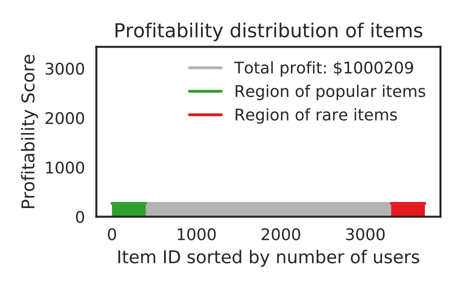
A single universal ranking is suboptimal.

Popularity is the optimal non-personalization algorithm.

Profitability Score Metric

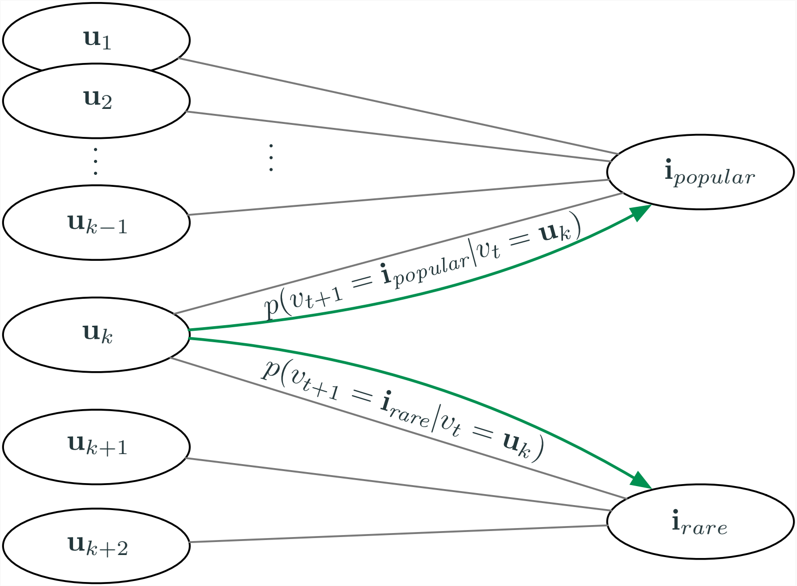


Precision Score Distribution

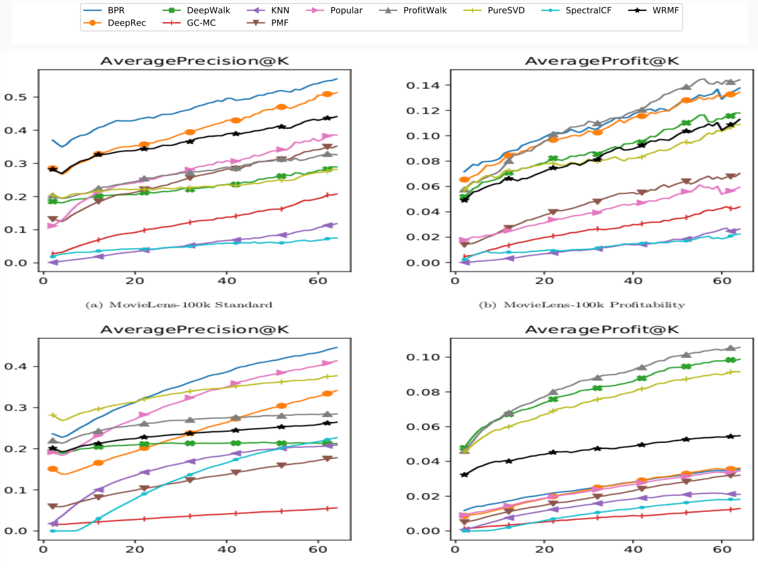


Profit Score Distribution

ProfitWalk


$\mathbf{u}_1$
$\mathbf{u}_2$
$\mathbf{u}_{k-1}$
$\mathbf{u}_k$
$\mathbf{u}_{k+1}$
$\mathbf{u}_{k+2}$
$\mathbf{i}_{popular}$
$\mathbf{i}_{rare}$
$p(v_{t+1} = \mathbf{i}_{popular} | v_t = \mathbf{u}_k)$
$p(v_{t+1} = \mathbf{i}_{rare} | v_t = \mathbf{u}_k)$

Results



(a) MovieLens-100k Standard

(b) MovieLens-100k Profitability

(c) MovieLens-1m Standard

(d) MovieLens-1m Profitability

Results

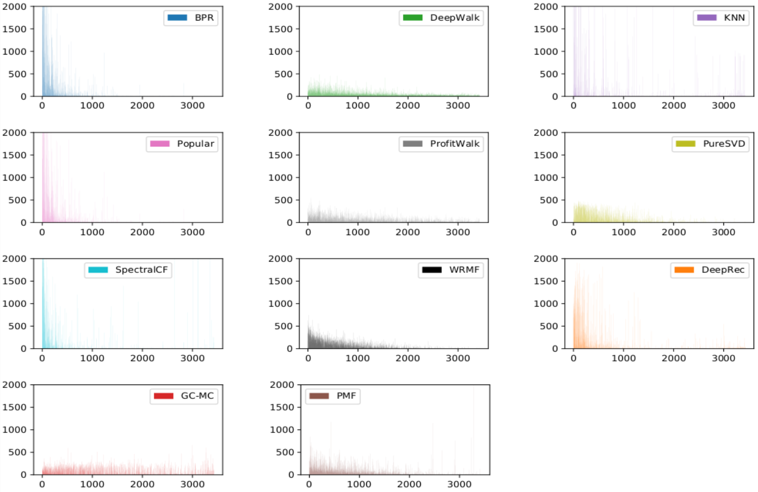
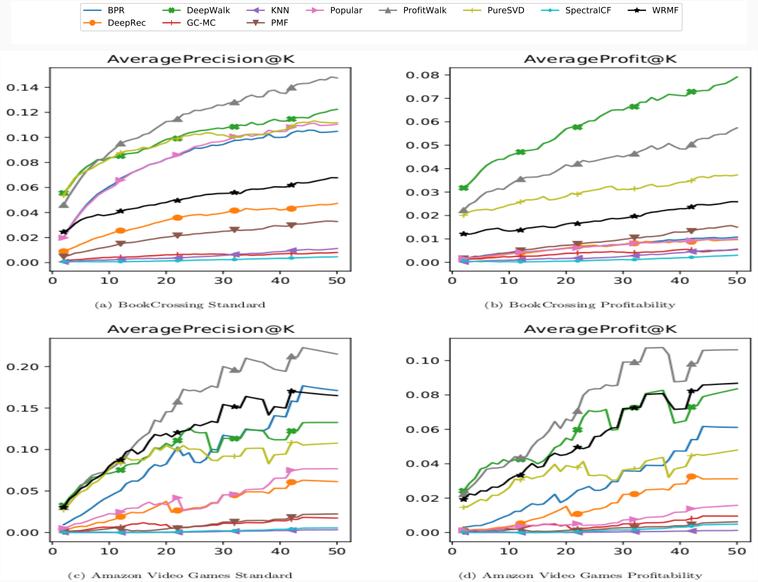


Figure 7.6: Recommendation Popularity of Algorithms on MovieLens-1m

Results


BPR
DeepRec
DeepWalk
GC-MC
KNN
PMF
Popular
ProfitWalk
PureSVD
SpectralCF
WRMF
AveragePrecision@K
AverageProfit@K
(a) BookCrossing Standard
(b) BookCrossing Profitability
(c) Amazon Video Games Standard
(d) Amazon Video Games Profitability

Results

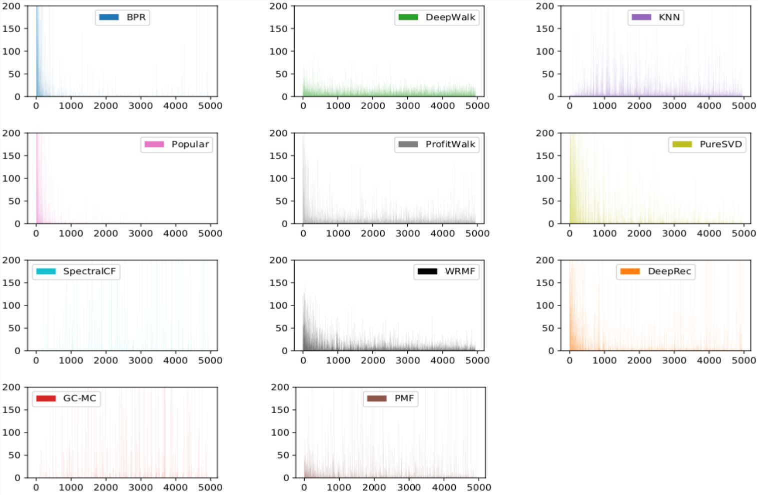


Figure 7.7: Recommendation Popularity of Algorithms on BookCrossing

Conclusion

Conclusion

Convolution-based Deep Graph Embeddings:

- under-performs under recommender system's analysis.

SGNS-based Deep Graph Embeddings:

- has great embeddings, respects metric spaces and is able to predict categorical information.
- outperforms on sparse datasets, especially on the long-tail.
- can be extended to optimize for the long-tail.
- has better popularity distribution.

Bias training of existing models towards profitability.
New metric that encourages profitable recommendations.

Questions

Appendix

Identity Similarity: An item should be most similar to itself.

$$Identity\ Similarity = \frac{1}{|I|}\sum_{i=1}^{|I|} 1[\underset{j}{\operatorname{argmax}}(W_i^l \cdot W_j^l) == i] \tag{1}$$

Appendix

Cluster Similarity: Average of similarity within a cluster centered at an item should be similar relative to that item's similarity against its K nearest neighbours similarity score.

$$
\begin{aligned}
Cluster\ Similarity@K &= \frac{1}{|I|}\sum_{i=1}^{|I|}\frac{within_i@K}{center_i@K} \\
within_i@K &= \frac{2}{K(K-1)}\sum_{j=1}^{K-1}\sum_{l=j+1}^{K} W_j^l \cdot W_l^l \\
center_i@K &= \frac{1}{K}\sum_{j=1}^{K} W_i^l \cdot W_j^l \\
I_j, I_k &\in I_i^{KNN}
\end{aligned}
\tag{1}
$$

Appendix

Binary Similarity: K-Nearest neighbour predicts the correct label using Binary Cross Entropy.

$$
\begin{aligned}
Binary\ Similarity@K &= \frac{1}{|I|}\sum_{i=1}^{|I|}\sum_{c=1}^{|C_i|} P_i(class(i)=c)\ log_2(P_{I_i^{KNN}}(class(i)=c)) \\
P_i(class(i)=c) &= \frac{1+\alpha}{|C_i|(1+\alpha)} \\
P_{I_i^{KNN}}(class(i)=c) &= \frac{1}{K}\sum_{j\in I_i^{KNN}} \frac{1[c\in C_j]+\alpha}{|C_j|(1+\alpha)}
\end{aligned}
\tag{1}
$$

$\alpha = 10$ is a positive constant for numerical stability if there exist a class $c \in C_i$ where none of the neighbours of item i intersects.

Appendix

Average Similarity: Average of items in the same label should result in being closest to another item in the same label.

$$
\begin{aligned}
Average\ Similarity@K &= \frac{1}{|C|}\sum_{c \in C} ClassPrecision@K_c(I_c^{KNN}) \\
W_c^I &= \frac{1}{K}\sum_{k=1, i_k \in I_c}^{K} W_{i_k}^I
\end{aligned} \tag{1}
$$

Appendix

$$
\begin{aligned}
\text{HitItemSet@}K_u &= \hat{r}_{u,:K} \cap r^{test}_{u,:} \\
\text{HitScore@}K_u &= \sum_{i \in \text{HitItemSet@}K_u} s_i \\
\text{OptimalScore@}K_u &= \sum_{i \in r^{test}_{u,:K}} s_i \\
\text{Precision@}K_u &= \frac{\text{HitScore@}K_u}{\text{OptimalScore@}K_u} \\
\text{AP@}K_u = \text{AveragePrecision@}K_u &= \frac{1}{K} \sum_{k=1}^{K} \text{Precision@}k_u \\
\text{mAP@}K = \text{AveragePrecision@}K &= \frac{1}{|U^{test}|} \sum_{u \in U^{test}} \text{AP@}K_u \\
\text{HitScore}_i = \sum_{u \in r^{test}_{:,i}} s_i &= |r^{test}_{:,i}| \times s_i, \text{TotalScore} = \sum_{i \in I^{test}} \text{HitScore}_i
\end{aligned}
\tag{1}
$$

Comparison between Standard and Profitability

Properties	Standard	Profitability
$\textbf{score}_i = s_i$	1.0	$\frac{1}{\lvert r^{test}_{:,i}\rvert} * (\frac{\lvert R^{test}\rvert}{\lvert I^{test}\rvert})$
$\textbf{HitScore@}K_u$	$\lvert\textbf{HitItemSet@}K_u\rvert$	as defined above
$\textbf{OptimalScore@}K_u$	K	as defined above
$\textbf{HitScore}_i$	$\lvert r^{test}_{:,i}\rvert$	$\frac{\lvert R^{test}\rvert}{\lvert I^{test}\rvert} = constant$
TotalScore	$\lvert R^{test}\rvert$	$\lvert R^{test}\rvert$
$range(\textbf{mAP@}K)$	[0.0, 1.0]	[0.0, 1.0]

Future Work

Future Work

- Develop embeddings that explicitly respect metric spaces.
- Develop embeddings that are robust towards adversarial attacks.
- Remove bias towards sensitive attributes from embeddings.

References i

[1] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen.
Challenging the long tail recommendation.
Proceedings of the VLDB Endowment, 5(9):896–907, 2012.