1、概述

这篇 RecSys 2019 的 Best Paper，从标题看来就很强，有种“在座各位都是……”的感觉。这两天通读下来，个人认为论文的贡献主要还是在于对学界敲响警钟，而不在于 new idea。

总的来说，作者从2018年的顶会里挑选了18篇（RecSys:7, KDD:4, WWW:4, SIGIR:3）深度学习Top-N推荐模型的文章，发现其中只有7篇（RecSys:1, KDD:3, WWW:2, SIGIR:1）的结果能被不那么难的进行复现，而这7篇之中有6篇是往往能被相对简单的算法超越。剩下的一篇确实能显著的超越baseline，但并不总能超越非神经网络的线性排序算法。

毕竟通篇强调 Reproducibility，作者当然有将实验代码公开到 GitHub 上：https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation ，有兴趣的同学可以进一步研究。

2、Baseline

作为参考，论文选择了以下7个算法作为 Baseline 算法。

2.1 TopPopular

顾名思义，直接取热门商品种的Top-N。其中热门程度通过显性或隐性的打分数量。

2.2 ItemKNN

传统的基于kNN与物品间相似度的协同过滤算法。

用$\overrightarrow{r_i}, \overrightarrow{r_j} \in R^{\|U\|}$分别表示物品$i, j$的打分向量，向量维度$\|U\|$表示用户总数；则物品间相似度可通过余弦相似度进行计算：

$$s_{ij} = \frac{\overrightarrow{r_i}\overrightarrow{r_j}}{||\overrightarrow{r_i}|| * ||\overrightarrow{r_j}|| + h}$$

其中，打分向量可选用 TF-IDF 或者 BM25 进行加权；相似度也可不用余弦相似度，而直接用向量内积。

得到物品间相似度以后，根据用户所浏览过的物品找到相似的物品即可。

2.3 UserKNN

传统的基于kNN与用户间相似度的协同过滤算法。

整体流程与 ItemKNN 相似。用$\overrightarrow{r_i}, \overrightarrow{r_j} \in R^{\|I\|}$分别表示用户$i, j$的打分向量，向量维度$\|I\|$表示物品总数；则用户间相似度可通过余弦相似度进行计算：

$$s_{ij} = \frac{\overrightarrow{r_i}\overrightarrow{r_j}}{||\overrightarrow{r_i}|| * ||\overrightarrow{r_j}|| + h}$$

同样的，打分向量可选用 TF-IDF 或者 BM25 进行加权；相似度也可不用余弦相似度，而直接用向量内积。

得到用户间相似度以后，根据用户找到与其相似的用户，推送相似用户所浏览过的物品即可。

2.4 ItemKNN-CBF

与 ItemKNN 基本一致，只是不是简单的使用的物品的打分向量，而是采用基于物品content计算物品的特征向量。

用$\overrightarrow{f_i}, \overrightarrow{f_j} \in R^{\|F\|}$分别表示物品$i, j$的特征向量，向量维度$\|F\|$表示特征总数；则物品间相似度可通过余弦相似度进行计算：

$$s_{ij} = \frac{\overrightarrow{f_i}\overrightarrow{f_j}}{||\overrightarrow{f_i}|| * ||\overrightarrow{f_j}|| + h}$$

2.5 ItemKNN-CFCBF

结合 ItemKNN 与 ItemKN-CBF ，将$\overrightarrow{r_i}$与$\overrightarrow{f_i}$简单拼接成$\overrightarrow{v_i} = [\overrightarrow{r_i}, w\overrightarrow{f_i}]$。后面的步骤一样，不再赘述。

2.6 $P^3\alpha$

基于图上random-walk的思想。

用r_{ui}表示用户u对物品i的打分，N_u表示用户u打分的总数，N_i表示物品i打分的总数。物品相似度计算如下：

$$\begin{aligned} s_{ij} &= \sum_u p_{ju} * p_{ui} \\ &= \sum_u (\frac{r_{uj}}{N_j})^\alpha * (\frac{r_{ui}}{N_u})^\alpha \end{aligned}$$

后面的步骤则与 ItemKNN 的一样。

2.7 $RP^3\beta$

在 $P^3\alpha$ 的基础上进行改进。假设物品的热门度为h_i。（存疑？）则物品相似度改为：

$$s_{ij} = (\sum_u (\frac{r_{uj}}{N_j})^\alpha * (\frac{r_{ui}}{N_u})^\alpha) / (q_i^\beta * q_j^\beta)$$

3、实验

对于所有的Baseline算法，统一利用 Scikit-Optimize 通过 Bayesian Search 自动找到最优参数。k取5~800，h取0~1000，α, β取0~2。

3.1 Collaborative Memory Networks (CMN)

> Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In Proceedings SIGIR '18. 515–524.

Table 2: Experimental results for the CMN method using the metrics and cutoffs reported in the original paper. Numbers are printed in bold when they correspond to the best result or when a baseline outperformed CMN.

	CiteULike-a			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1803	0.1220	0.2783	0.1535
UserKNN	**0.8213**	**0.7033**	**0.8935**	**0.7268**
ItemKNN	**0.8116**	**0.6939**	0.8878	**0.7187**
$P^3\alpha$	**0.8202**	**0.7061**	0.8901	**0.7289**
$RP^3\beta$	**0.8226**	**0.7114**	**0.8941**	**0.7347**
CMN	0.8069	0.6666	0.8910	0.6942
	Pinterest			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1668	0.1066	0.2745	0.1411
UserKNN	**0.6886**	**0.4936**	0.8527	**0.5470**
ItemKNN	**0.6966**	**0.4994**	**0.8647**	**0.5542**
$P^3\alpha$	0.6871	**0.4935**	0.8449	**0.5450**
$RP^3\beta$	**0.7018**	**0.5041**	**0.8644**	**0.5571**
CMN	0.6872	0.4883	0.8549	0.5430
	Epinions			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	**0.5429**	**0.4153**	**0.6644**	**0.4547**
UserKNN	0.3506	0.2983	0.3922	0.3117
ItemKNN	0.3821	0.3165	0.4372	0.3343
$P^3\alpha$	0.3510	0.2989	0.3891	0.3112
$RP^3\beta$	0.3511	0.2980	0.3892	0.3103
CMN	0.4195	0.3346	0.4953	0.3592

3.2 Metapath based Context for RECommendation (MCRec)

> Binbin Hu, Chuan Shi,Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging metapath based context for top-n recommendation with a neural co-attention model. In Proceedings KDD '18. 1531–1540.

Table 3: Comparing MCRec against our baselines (MovieLens100k)

	PREC@10	REC@10	NDCG@10
TopPopular	0.1907	0.1180	0.1361
UserKNN	0.2913	0.1802	0.2055
ItemKNN	**0.3327**	**0.2199**	**0.2603**
$P^3\alpha$	0.2137	0.1585	0.1838
$RP^3\beta$	0.2357	0.1684	0.1923
MCRec	0.3077	0.2061	0.2363

3.3 Collaborative Variational Autoencoder (CVAE)

> Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In Proceedings KDD '17. 305–314.

Table 4: Experimental results for CVAE (CiteULike-a).

	REC@50	REC@100	REC@300
TopPopular	0.0044	0.0081	0.0258
UserKNN	0.0683	0.1016	0.1685
ItemKNN	**0.0788**	0.1153	0.1823
$P^3\alpha$	**0.0788**	0.1151	0.1784
$RP^3\beta$	**0.0811**	0.1184	0.1799
ItemKNN-CFCBF	**0.1837**	**0.2777**	**0.4486**
CVAE	0.0772	0.1548	0.3602

3.4 Collaborative Deep Learning (CDL)

> HaoWang, NaiyanWang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In Proceedings KDD '15. 1235–1244.

Table 5: Experimental results for CDL on the dense *CiteULike-a* dataset.

	REC@50	REC@100	REC@300
TopPopular	0.0038	0.0073	0.0258
UserKNN	**0.0685**	0.1028	0.1710
ItemKNN	**0.0846**	**0.1213**	0.1861
$P^3\alpha$	**0.0718**	**0.1079**	0.1777
$RP^3\beta$	**0.0800**	**0.1167**	0.1815
ItemKNN-CBF	**0.2135**	**0.3038**	**0.4707**
ItemKNN-CFCBF	**0.1945**	**0.2896**	**0.4620**
CDL	0.0543	0.1035	0.2627

3.5 Neural Collaborative Filtering (NCF)

> Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings WWW '17. 173–182.

Table 6: Experimental results for NCF.

	Pinterest			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.1663	0.1065	0.2744	0.1412
UserKNN	0.7001	**0.5033**	0.8610	**0.5557**
ItemKNN	**0.7100**	**0.5092**	**0.8744**	**0.5629**
$P^3\alpha$	0.7008	**0.5018**	0.8667	**0.5559**
$RP^3\beta$	**0.7105**	**0.5116**	**0.8740**	**0.5650**
NeuMF	0.7024	0.4983	0.8719	0.5536
	Movielens 1M			
	HR@5	NDCG@5	HR@10	NDCG@10
TopPopular	0.3043	0.2062	0.4531	0.2542
UserKNN	0.4916	0.3328	0.6705	0.3908
ItemKNN	0.4829	0.3328	0.6596	0.3900
$P^3\alpha$	0.4811	0.3331	0.6464	0.3867
$RP^3\beta$	0.4922	0.3409	0.6715	0.3991
NeuMF	0.5486	0.3840	0.7120	0.4369
SLIM	**0.5589**	**0.3961**	**0.7161**	**0.4470**

3.6 Spectral Collaborative Filtering (SpectralCF)

> Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral Collaborative Filtering. In Proceedings RecSys '18. 311–319.

Table 7: Experimental results for SpectralCF (MovieLens1M, using own random splits and five repeated measurements).

	Cutoff 20		Cutoff 60		Cutoff 100	
	REC	MAP	REC	MAP	REC	MAP
TopPopular	**0.1853**	**0.0576**	**0.3335**	**0.0659**	**0.4244**	**0.0696**
UserKNN CF	**0.2881**	**0.1106**	**0.4780**	**0.1238**	**0.5790**	**0.1290**
ItemKNN CF	**0.2819**	**0.1059**	**0.4712**	**0.1190**	**0.5737**	**0.1243**
$P^3\alpha$	**0.2853**	**0.1051**	**0.4808**	**0.1195**	**0.5760**	**0.1248**
$RP^3\beta$	**0.2910**	**0.1088**	**0.4882**	**0.1233**	**0.5884**	**0.1288**
SpectralCF	0.1843	0.0539	0.3274	0.0618	0.4254	0.0656

3.7 Variational Autoencoders for Collaborative Filtering (Mult-VAE)

> Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In Proceedings WWW '18. 689–698.

Table 8: Experimental results for Mult-VAE (Netflix data), using metrics and cutoffs reported in the original paper.

	REC@20	REC@50	NDCG@100
TopPop	0.0782	0.1643	0.1570
ItemKNN CF	0.2088	0.3386	0.3086
$P^3\alpha$	0.1977	0.3346	0.2967
$RP^3\beta$	0.2196	0.3560	0.3246
SLIM	0.2551	0.3995	0.3745
Mult-VAE	**0.2626**	**0.4138**	**0.3756**

Table 9: Experimental results for Mult-VAE using additional cutoff lengths for the Netflix dataset.

	NDCG@20	NDCG@50	REC@100	NDCG@100
SLIM	**0.2473**	**0.3196**	0.5289	0.3745
Mult-VAE	0.2448	0.3192	**0.5476**	**0.3756**

结语

这里先对论文进行简单的总结与摘要。文中提到的七篇论文的对比实验.....光看数据也没有什么收获，得等后面具体的看七篇论文后再展开来说了。