【序列推荐】RecSys2020|FISSA---融合物品相似度模型和自注意力网络的推荐

原创 潜心 推荐算法的小齿轮 2020-12-23

收录于话题

#序列推荐 11 #召回 9 #推荐系统论文与复现 22

FISSA: Fusing Item Similarity Models with Self-Attention Networks for Sequential Recommendation

Jing Lin
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
linjing2018@email.szu.edu.cn

Weike Pan*
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
panweike@szu.edu.cn

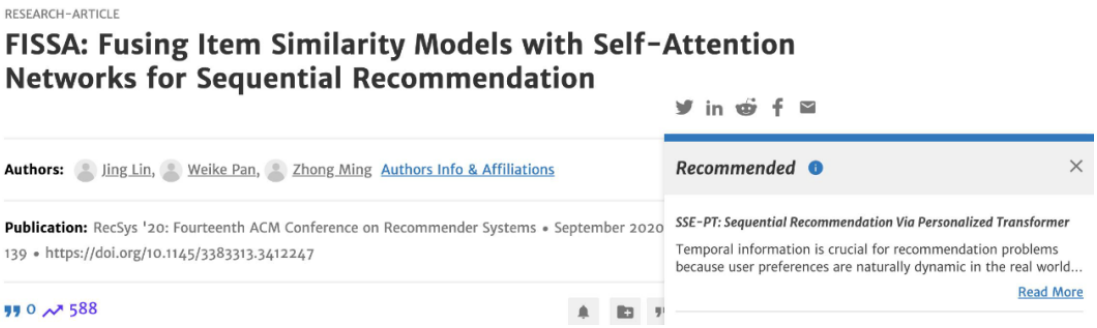
Zhong Ming*
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
mingz@szu.edu.cn

前言

文章发表在2020年的顶会RecSys，提出了一个融合物品相似度和自注意力机制的序列推荐模型FISSA。FISSA融入了SASRec模型，将其看作是提取用户行为的局部表示（local representation，短期兴趣），又加入了对全局偏好（global preferences，长期兴趣）的提取。并且，提出一种类似LSTM单元的门控函数，将两者进行有效的融合。

「原文地址」：https://dl.acm.org/doi/10.1145/3383313.3412247

有意思的是，给我推荐了上次发表在RecSys的论文SSE-PT。



本文约3.0k字，预计阅读10分钟。

概要

本篇论文抛出了当下序列推荐模型中两个重要的问题：

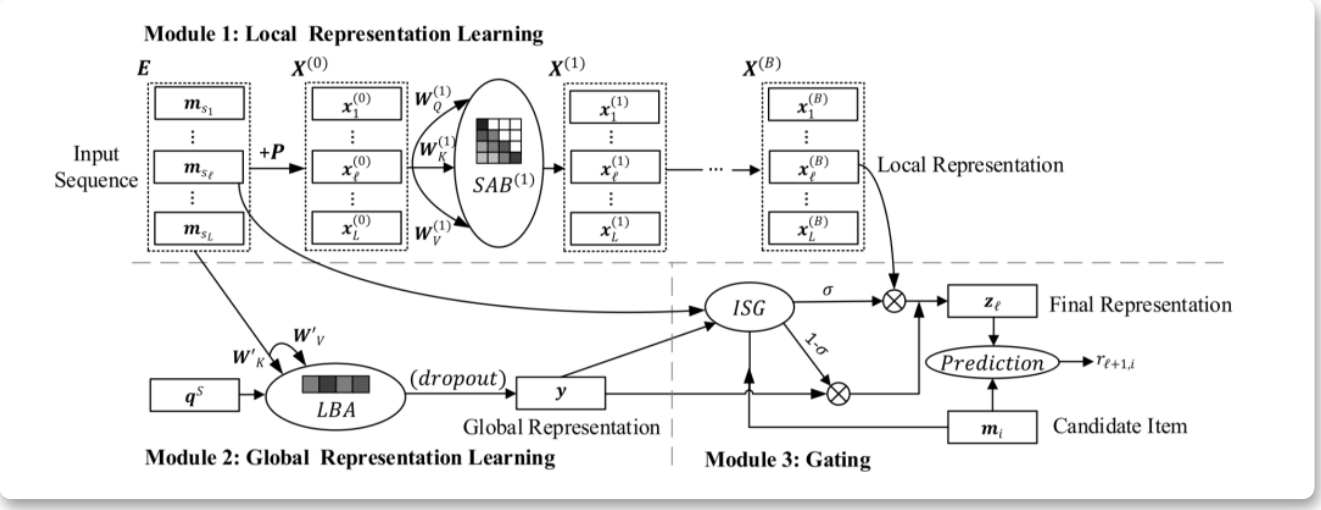
1. 对于很多序列推荐模型中，例如SASRec、Caser，它们太过关注于用户的局部偏好，也就是短期兴趣，而忽略了用户的长期兴趣。而向AttRec、STAMP等模型，建立长期兴趣（作者将其称为全局、静态的偏好）的方法太过简单，低估了其作用；
2. 作者认为，现在绝大多数模型，都认为能够从用户的历史行为中完全能够捕捉到用户当前兴趣，这是一种理想主义。在现实中，用户的意图往往是很难进行确定的，特别是处于一个长期行为（电影、长时间的一个购物记录），而不是一个短期行为（听音乐）。判断一个新产品是否能吸引用户的正确方法是，考虑它如何能引起用户不同部分的兴趣(即短期兴趣和长期兴趣)。

【注】谈一谈我个人的一些看法，关于第1点，其实现在很多序列推荐模型都开始将用户行为序列建模分为两个部分：短期偏好和长期偏好，但是对于长期偏好的建模确实过于简单【尝试过通过 `self-attention` 来提取长期偏好（平均），不过效果一般，短期偏好才是yyds】。第2点，等于是得到两个兴趣后，如何去合并？不错的方法是分配不同的权重，而权重是通过计算候选物品与兴趣的相似程度。其实DIN模型就是这样做的（通过attention），但是他是CTR模型，如果放到召回（Top-K）中，会影响模型预测效率，应该很难使用FAISS库等可以计算邻近。

因此，作者提出了FISSA模型有效的解决上述两个问题。

模型结构

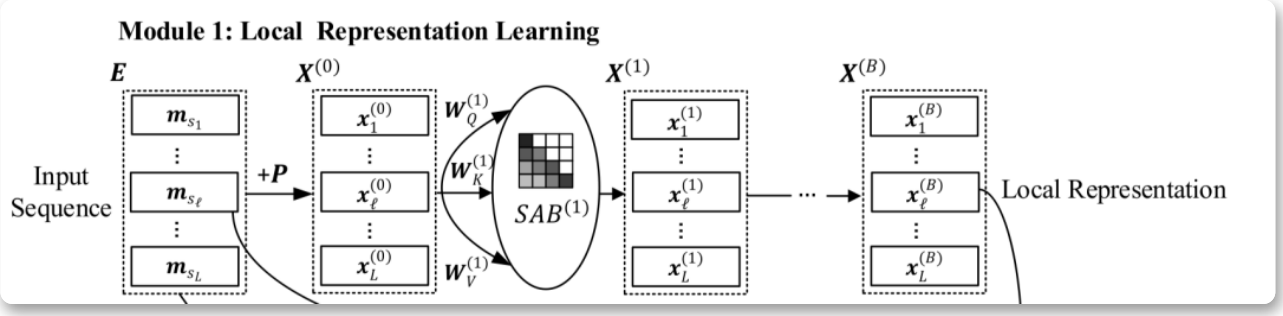
FISSA模型结构清晰，分为三个模块，局部表示学习（Local Representation Learning）、全局表示学习（Global Representation Learning）、门控函数（Gating）。模型结构如下所示：



问题定义

定义用户集合$\mathcal{U}$和物品集合$\mathcal{I}$，对于单个用户u，其历史物品序列为。序列推荐的目的是，模型能够为用户u提供一个推荐列表，下一个真实的交互物品$s^u_{|S^u|+1} \in I \setminus \mathcal{S}^u$出现在推荐列表中，并且排名尽可能得高。

局部表示学习



第一部分是采用 self-attention 的局部表示学习模块，并没有什么创新，不过作者自己也指出了，本篇论文不要过多的关注局部表示模块，因为大家都已经做得很好了，所以主要来看全局表示学习是如何做的。这里我简单阐述下：

作者规定输入序列为用户u的最近前L个行为（不足进行填充），$\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$为物品的embedding矩阵，$d$为embedding维度。因此，用户行为的Embedding矩阵为$\boldsymbol{E} = [\boldsymbol{m}_{s_1}; \boldsymbol{m}_{s_2}; \ldots; \boldsymbol{m}_{s_L}] \in \mathbb{R}^{L \times d}$。采用 self-attention 网络进行局部表示的学习。由于注意力网络不具备位置关系，因此添加了可学习的位置embedding矩阵$\boldsymbol{P} = [p_1; p_2; \ldots; p_L] \in \mathbb{R}^{L \times d}$，得到输入矩阵$\boldsymbol{X}^{(0)} = [x_1; x_2; \ldots; x_L] \in \mathbb{R}^{L \times d}$：

$$\boldsymbol{x}_\ell^{(0)} = \boldsymbol{m}_{s_\ell} + \boldsymbol{p}_\ell, \ell \in \{1, 2, \ldots, L\}$$

【注】与Transformer提供的方法不同，但其实在很多使用self-attention网络的序列推荐模型中都使用上述方法，我经过实验，确实比Transformer提供的positional encoding方法效果更好。

接下来就是将通过多个 self-attention block(SABs)：

$$\boldsymbol{X}^{(b)} = SAB^{(b)}\left(\boldsymbol{X}^{(b-1)}\right), b \in \{1, 2, \ldots, B\}$$

对于单个 SAB：

$$SAB(X) = FFL(SAL(X))$$

$$X' = SAL(X) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)\Delta \cdot V$$

$$FFL(X') = \text{ReLU}\left(X'W_1 + 1^T b_1\right)W_2 + 1^T b_2$$

其中FFL为前向传播网络。

最终，通过多个 SABs ，其中输出向量$\boldsymbol{x}_{\ell}^{(B)} \in \mathbb{R}^{1\times d}$，这代表第$l$步的用户动态偏好。所以整体得到的是$X^(B) \in \mathbb{R}^{L\times d}$

【注】这里稍微有点歧义，SASRec也是类似的写法，看过两篇论文以及他们的源代码我才能真正理解为啥这里输出的是一个向量。我简单进行下解释：

> 首先，SASRec、FISSA的输入是一个用户的整个历史行为，即在训练中能获得的长度为L的序列。并不是像一些文章定义的那样，对于单个用户一整串的行为，进行切分，例如用户有历史行为{1，2，3，4，5}，规定长度$L=3$，那么将其划分为{2}、{2，3}、{2，3，4}，标签}、{3}、{4}、{5}。这种形式表达比较清楚，但是实际实验的时候，会消耗很多不必要的内存。对于FISSA来说，他们的输入就是{2，3，4}，但是其标签为{ 3，4，5}，通过掩码的方式同样能达到实验目的。

因此，对于通过 SABs 得到$X^{(b)} \in \mathbb{R}^{L\times d}$，而对于每个时间步，提取的是该时间步输出的向量，即上述的$\boldsymbol{x}_{\ell}^{(B)}$。对于划分多个样本的情况，那么此时提取的是$X^{(b)} \in \mathbb{R}^{L\times d}$的**「最后一个向量」**。对比发现，FISSA/SASRec采用的方式减少了内存的消耗，且提高了训练速度。

关于 SABs 的数量，作者也提到了，如果为1，那么可以来捕获长期依赖，但是对于2及以上，则更关注于最近的信息。

全局表示学习

第二部分是全局表示学习模块，用来提取用户的长期偏好，作者先是通过FISM（传统的推荐模型）来引入。在FISM中，用户u对下一个交互物品$s_{\ell+1}^{u}$的偏好被建立为其他交互物品表示的聚合：

$$\tilde{\boldsymbol{y}}_{\ell+1}^{u}=\frac{1}{\sqrt{\left|\mathcal{S}^{u} \setminus\left\{s_{\ell+1}^{u}\right\}\right|}} \sum_{i^{\prime} \in \mathcal{S}^{u} \setminus\left\{s_{\ell+1}^{u}\right\}} \boldsymbol{m}_{i^{\prime}}$$

对下一个交互物品的预测分数为：$r_{\ell+1, s_{\ell+1}^{u}}^{u}=\tilde{\boldsymbol{y}}_{\ell+1}^{u} \boldsymbol{m}_{s_{\ell+1}^{u}}^{T}$，这可以理解为用户$u$的历史行为与下一个物品$s_{\ell+1}^{u}$的相似度。

作者认为不同的历史物品应该具有不同的权重。但如何来分配权重呢？这里作者采用了 attntion mechanism，但是比较有意思的是模型中的 query 是可学习的$\boldsymbol{q}^{S} \in \mathbb{R}^{1 \times d}$，通过 query 来实现对每个历史物品权重的分配（也就是attention网络，作者将其称为 location-based attention layer LBA(·)）：

$$\boldsymbol{y}=LBA(\boldsymbol{E})=\operatorname{softmax}\left(\boldsymbol{q}^{S}\left(\boldsymbol{E} \boldsymbol{W}_{K}^{\prime}\right)^{T}\right) \boldsymbol{E} \boldsymbol{W}_{V}^{\prime}$$

其中$\boldsymbol{E} \in \mathbb{R}^{L \times d}$为初始输入矩阵，$\boldsymbol{W}_{K}^{\prime}, \boldsymbol{W}_{V}^{\prime} \in \mathbb{R}^{d \times d}$为学习矩阵。

另外，作者加入了 dropout layer：

$$\boldsymbol{y}_{\ell}=\operatorname{Dropout}(\boldsymbol{y}), \ell \in\{1,2, \ldots, L\}$$

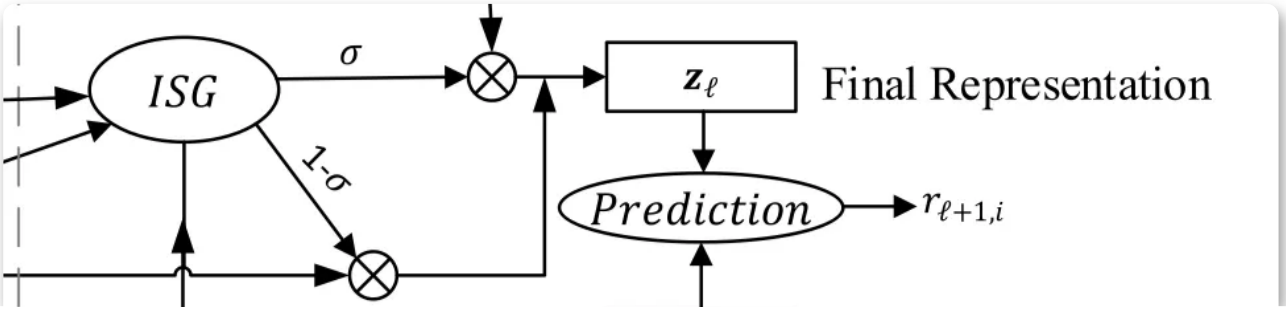
因此最终得到的全局表示矩阵$\mathbf{Y} \in \mathbb{R}^{L \times d}$。

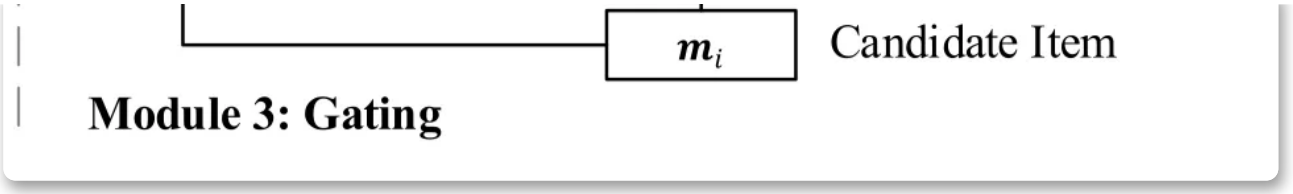
【注】这里得到$\mathbf{Y}$的原因上述解释相同。

联合---物品相似度门控

第三部分是局部表示与全局表示的融合。如何联合局部表示和全局表示，作者早期分别做了求和（summation）和拼接（concatenation）的测试，结果证明求和效果更好，但作者认为这些组合方法仍然只是基于历史信息，这可能是理想化的。

为了解决序列推荐中用户意图不确定性的问题，受**「神经注意物品相似度(NAIS)」** 的启发，作者提出一个 item similarity gating 模块，通过建模候选物品i和最近交互的物品s_{ℓ}之间的物品相似度，以及候选物品i和历史行为物品的聚合之间的物品相似度来计算局部表示和全局表示的权重，如下所示：





作者有如下定义：

1. 门控函数g的输出作为局部表示的权重，且$0
2. Gating模块中，输入有三种 item ，即候选物品i，最近交互物品s_ℓ（局部表示）和历史物品$i' \in \mathcal{S}^u$的聚合（全局表示），结合计算得到两种物品相似度【上述提到】

门控函数g的定义如下：

$$g = \sigma(\mathrm{ISG}(\boldsymbol{m}_{s_\ell}, \boldsymbol{y}, \boldsymbol{m}_i)) = \sigma([\boldsymbol{m}_{s_\ell}, \boldsymbol{y}, \boldsymbol{m}_i]\boldsymbol{W}_G + b_G)$$

其中$ISG(\cdot,\cdot,\cdot)$表示物品相似度的门控函数，$\boldsymbol{m}_{s_\ell}$表示最近历史行为的embedding向量，$\boldsymbol{y}$为全局表示，$\boldsymbol{m}_i$为候选物品embedding向量，$\boldsymbol{W}_G \in \mathbb{R}^{3d\times 1}$，$\sigma(\cdot)$为sigmoid激活函数。

【注】其实就是将三者进行拼接，然后经过一个全连接网络。然后g表示候选物品对局部表示的权重。

第l步序列的最终表示【用户表示】为

$$\boldsymbol{z}_\ell = \boldsymbol{x}_\ell^{(B)} \otimes g + \boldsymbol{y} \otimes (1-g)$$

其中$\otimes$表示逐个元素相乘。

最后，预测候选物品i为第$l+1$个用户交互的得分为：

$$r_{\ell+1,i} = z_\ell (\boldsymbol{m}_i)^T$$

损失函数：

$$\mathcal{L} = -\sum_{u\in\mathcal{U}}\sum_{\ell=1}^{L-1}\delta(s^u_{\ell+1})\left[\log\left(\sigma\left(r^u_{\ell+1,s^u_{\ell+1}}\right)\right) + \log\left(1-\sigma\left(r^u_{\ell+1,j}\right)\right)\right]$$

【注】与其他直接计算用户抽象表示与候选物品的相似度不同的是，多了一个门控函数，但是却采用了类似attention的做法，不过只有两个值（全局和局部），不清楚这样做在工业界是否可行？

实验

「数据集：」

Dataset	# Users	# Items	# Interactions	Avg. Length	Density
Beauty	40,226	54,542	353,962	8.80	0.02%
Games	29,341	23,464	280,945	9.58	0.04%
Steam	281,428	13,044	3,488,899	12.40	0.10%
Foursquare	22,748	11,146	145,106	6.38	0.06%
Tmall	201,139	97,636	1,936,790	9.63	0.01%

Table 1: Statistics of the processed datasets.

「指标：」

- recall（Rec@10）：在用户的单物品召回测试中，等价于点击率hit ratio；
- normalized discounted cumulative gain (NDCG@10);

验证时，与SASRec一样，正负样本比例为1：100。

「实验结果：」

Dataset	Metric	MF-based				DL-based					FISSA vs.
		BPRMF	FISM	FPMC	Fossil	GRU4Rec+	Caser	SASRec	CAR	FISSA	SASRec
Beauty	Rec@10	0.2498	0.3533	0.2397	0.3570	0.2729	0.2809	0.3609	0.3660	**0.4164**	15.38%
	NDCG@10	0.1148	0.1942	0.1093	0.2108	0.1683	0.1610	0.2173	0.2214	**0.2484**	14.35%
Games	Rec@10	0.3454	0.4791	0.3665	0.4800	0.4825	0.4810	0.6009	0.5947	**0.6743**	12.22%
	NDCG@10	0.1981	0.2631	0.2115	0.2708	0.2906	0.2857	0.3685	0.3644	**0.4134**	12.19%
Steam	Rec@10	0.1023	0.3183	0.1735	0.2926	0.3177	0.2686	0.3886	0.3927	**0.4294**	11.79%
	NDCG@10	0.0468	0.1703	0.0849	0.1546	0.1707	0.1342	0.2144	0.2164	**0.2420**	13.91%
Foursquare	Rec@10	0.2659	0.3977	0.3641	0.4223	0.4324	0.4043	0.4808	0.4868	**0.5106**	6.20%
	NDCG@10	0.1287	0.2025	0.1873	0.2303	0.2375	0.2108	0.2611	0.2629	**0.2794**	7.04%
Tmall	Rec@10	0.1744	0.2149	0.1739	0.2303	0.3526	0.2813	0.4204	0.4184	**0.4412**	4.94%
	NDCG@10	0.0825	0.1043	0.0821	0.1142	0.2072	0.1531	0.2385	0.2380	**0.2451**	2.78%

Table 2: Recommendation performance of our FISSA and eight baselines on five datasets.

FISSA取得了最好的实验结果，特别是在Beauty、Games、Steam三个数据集上更为明显。详细的实验内容可以参考原文。

总结

读完这篇文章还是有一定收获的，作者很多的想法和我目前的一些idea类似，不过作者想得更加的全面（不然我也能发顶会了）。对于门控函数部分，我持保留意见，认为这样模型更接近于CTR模型，且会降低模型的效率，需要进行咨询下工业界的大佬进行更进一步的了解。