基于融合模型和后处理策略的公平性推荐算法

（KDD Cup 2020 Challenges for Modern E-Commerce Platform: Debiasing 技术报告）

王文
华东师范大学
51164500120@stu.ecnu.edu.cn

梁文伟
华东师范大学
51174500033@stu.ecnu.edu.cn

张伟
华东师范大学
zhangwei.thu2011@gmail.com

摘要

这篇文章主要描述我们队伍（ECNU@KDDCUP20）在 KDD Cup 2020 Challenges for Modern E-Commerce Platform: Debiasing 竞赛中采用的算法。我们的算法主要是基于多个深度学习框架计算出每个候选物品的得分。为了达到推荐的公平性，我们设计两个后处理策略来降低流行度较高的物品的得分和提高流行度较低的物品的得分。最后通过线性加权和的方式融合多个模型的得分得到推荐列表。该算法在竞赛中获得了第五名。

1 算法描述

1.1 问题定义

已知用户 u、用户历史点击过的 n 个物品 $S = \{e_1, e_2, \ldots, e_n\}$，点击 n 个物品的时间 $T = \{t_1, t_2, \ldots, t_n\}$ 以及下一个点击物品的时间 t_{n+1}，模型学习并预测用户的下一个点击的物品 e_{n+1}，生成最为可能的 50 个物品作为推荐列表，同时尽可能提高每个推荐的物品的公平性。

1.2 模型

我们的算法融合了四个深度学习模型的推荐结果。这四个模型通过两个模型框架（模型 1：**Attention 模型**，模型 2：**LastItem 模型**）分别通过两种不同的初始化方式训练得到。接下来，我们首先介绍模型的基础框架，然后详细介绍每个模型的计算细节。

1.2.1 模型基础框架

与现有的基于序列的推荐系统框架类似，我们的模型框架也分为一下几个步骤：1). 把每个用户和每个物品都映射成一个 k 维的向量；2). 基于历史点击序列计算出用户当前的表示，生成一个 k 维的向量；3). 用这个 k 维的用户向量表示与每个物品的 k 维向量表示计算相似度；4). 选取相似度得分最高的 50 个物品作为推荐列表。

具体的，对于每个用户 u，我们通过一个 Embedding 层把 u 映射为一个 k 维的向量 $\vec{\mathbf{u}}$：

$$\vec{\boldsymbol{u}} = Embedding_u(\mathrm{u}; \mathbf{F}),$$

对物品 e 也是类似：

$$\vec{\boldsymbol{e}} = Embedding_e(\mathrm{e}; \mathbf{E}),$$

F和E分别是用户的 embedding 矩阵和物品的 embedding 矩阵，它们的维度分别是用户数量×k 和物品数量×k。

模型的核心部分是根据用户 u 最近历史点击序列 S_u 计算出用户当前的表示 $\vec{\boldsymbol{h}}$：

$$\vec{\boldsymbol{h}} = \mathrm{model}(\vec{\boldsymbol{u}}, \mathrm{S}, \mathrm{T}; \theta),$$

其中θ是这一部分的模型参数。model函数在不同的模型中计算方式不一样，稍后我们会分别介绍我们采用的两个模型的model函数的具体细节。

为了得到每个物品的得分，我们计算用户表示和每个物品 embedding 的内积，物品 e 的得分为：

$$\mathrm{s}_i = \vec{\boldsymbol{h}} \cdot \vec{\boldsymbol{e}}$$

在模型训练阶段，我们使用 Adam 算法来最小化交叉熵损失函数来优化我们的模型。正样本就是用户下一个真正点击的物品；负样本应该是除去正样本之外的所有物品，但是由于物品的总数量太多，会导致训练效率太低，所以我们每次只随机挑选了 200 个物品当作负样本来训练。

1.2.2 模型 1：Attention 模型

因为用户的兴趣是会随着时间变化的，用户在较长时间之前点击的物品与最近点击的物品对用户兴趣的影响大小是不一样的。所以我们设计了一个考虑时间差的 attention 模型来自适应地学习不同用户在与当前不同时间差下点击的物品的权重。

我们首先通过下一个点击的时间 t_{n+1} 和用户历史点击的第 i 个物品的时间 t_i 计算出时间差 $\Delta \mathrm{t}_i = \mathrm{t}_{n+1} - \mathrm{t}_i$。然后按时间差大小分段，第 i 个物品对应的时间差分段 $d_i = \left\lfloor \ln \frac{\Delta \mathrm{t}_i}{\epsilon} \right\rfloor$，$\lfloor \cdot \rfloor$ 为向下取整，ε为一个时间缩放因子，在我们的实现中设为 5e4。然后我们映射第 i 个物品的重要度权重为 w_{d_i}，w 为一个可训练的向量，用 0 初始化。

最后，同时考虑用户 embedding 和物品加权的兴趣表示，该模型的用户表示的计算方式如下：

$$\mathrm{model}(\vec{\boldsymbol{u}}, \mathrm{S}, \mathrm{T}; \boldsymbol{w}) = \vec{\boldsymbol{u}} + \sum_{i=1}^{n} a_i * \vec{\boldsymbol{e}}_i,$$

其中 $a_i = \frac{\exp(w_{d_i})}{\sum_{j=1}^{n} \exp(w_{d_j})}$。

1.2.3 模型 2：LastItem 模型

因为用户最近点击的物品与用户下一个可能点击的物品是最相关的，所以在该模型中，我们简单地使用用户最近点击的物品的 embedding 作为用户当前的表示：

$$\text{model}(\vec{u}, \text{S}, \text{T}; \emptyset) = \vec{e}_n。$$

1.2.4 Embedding 初始化

在我们的实验中，我们对以上两个模型分别采用两种初始化策略来初始化物品的 embedding：随机初始化和用预训练内容表示初始化。值得注意的是：不管采用哪种方式初始化物品 embedding，用户 embedding 都是用随机初始化的。

随机初始化：物品的 embedding 用 TensorFlow 默认的初始化方法；

预训练内容表示初始化：预训练内容表示由官方提供的 128 维物品描述文本表示和 128 维物品图像表示，我们把这两个表示拼接起来变成一个 256 维的向量，用来初始化物品的 embedding。这里因为这个向量里面的值比较大，为了避免最后计算出的物品得分会太大影响训练，训练时候我们对最后的物品得分会再乘上 0.01。

2 后处理策略

常规的训练方式会使得模型偏好推荐流行度高的物品，因为如果某物品出现次数多，那么训练时候该物品作为正样本的概率大，就会使得模型给该物品预测出的得分会偏高，这样的话推荐列表中大部分都会是流行度较高的物品，出现推荐物品不公平的现象。而在本问题中，如何提升推荐的公平性是我们要考虑的方面。

为了避免推荐太多的高流行度物品，我们对物品的得分采用了几个后处理策略来平衡高流行度物品和低流行度物品的得分。

策略 1：为了减小高流行度物品的得分，我们对每个物品的得分 s 进行一个与物品流行度负相关的缩放：

$$\text{s} := s * \left(1 + \frac{1}{\ln(\text{p} + \text{e})}\right)$$

其中 p 为物品的流行度，既在训练集中出现次数，$\text{e} \approx 2.71828$为自然底数。

策略 2：因为最后评测的时候主要考虑低流行度物品预测效果，我们对低流行度的物品的得分再提高一下，考虑到低流行度物品也有流行度高低之分，我们应该对流行度较高的物品的得分提高比较小的幅度，而对流行度较低的物品的得分提高比较大的幅度，所以该策略的变换为：

$$\text{s} := s * \left(\alpha + \frac{1}{\text{p}}\right)$$

其中$\alpha \geq 1$为一个超参，在随机初始化的模型中设为 1.5，在用物品内容初始化的模型中设为 1.1。该策略仅对流行度较低的一半物品起作用。

3 模型融合

为了融合 4 个模型的结果，我们对每个物品在 4 个模型中的得分做一个加权和得到一个总的得分：

$$\text{final score} = \sum_{i=1}^{4} \text{v}_i * \text{s}_i$$

其中$\text{s}_{1\sim4}$分别为物品在 4 个模型（attention 模型随机初始化、attention 模型用内容表示初始化、LastItem 模型随机初始化、LastItem 模型用内容表示初始化）中的得分，我们通过一些实验尝试，把$\text{v}_{1\sim4}$分别设为 10、19、2.5、3。

3 实验

3.1 数据集

官方给定的数据主要包括三个部分：

1.用户个性化数据：用户年龄级别，用户的性别和用户所在的城市级别。因为这些特征数据缺失严重，我们没有使用这些特征。

2.预训练物品内容表示：预训练的物品描述文本表示 128 维，预训练的物品图像表示 128 维。

3.给定的用户点击数据，包含训练数据和测试数据。

3.2 数据预处理

原始数据是分了多个 phase，我们首先把多个 phase 的数据合并，去掉重复的记录。然后对每个 phase，我们随机挑选与测试用户同样数量的用户作为验证集用户，把验证集用户在相应 phase 的最后一个点击的物品当做验证数据。

值得注意的是，在模型预测推荐列表的时候，我们针对每个 phase 的用户，只预测在相应 phase 出现过的而且用户没有点击过的物品。

3.3 评价指标

对于实验的评价指标，我们采用的是 NDCG_50_full，NDCG_50_half，HitRate_50_full 以及 HitRate_50_half。我们采用官方提供的评测代码在线下计算验证集上的表现。

3.3 实现细节

我们采用 TensorFlow 实现和优化我们的模型，对于随机初始化的模型，我们设置向量维度 k 为 1024，学习率为 1e-4，attention 模型计算用户最近 3 个点击的物品。对于用物品内容初始化的模型，我们设置向量维度 k 为 256，学习率为 1e-3，attention 模型使用用户最近 5 个点击的物品。所有模型均采用 Adam 进行优化，batch size 设为 512。

3.4 实验结果

Figure 1：算法在 Track B 数据集上线下的结果。

策略	模型	hitrate_full	ndcg_full	hitrate_half	ndcg_half
原始模型	attention模型-random	0.6116	0.2674	0.3302	0.1214
	attention模型-content	0.5909	0.2439	0.3964	0.1346
	LastItem模型-random	0.4593	0.2030	0.2439	0.0896
	LastItem模型-content	0.4952	0.2092	0.3002	0.1054
策略1	attention模型-random	0.6267	0.2740	0.3988	0.1575
	attention模型-content	0.6073	0.2508	0.4937	0.1857
	LastItem模型-random	0.4649	0.2060	0.2759	0.1052
	LastItem模型-content	0.5160	0.2169	0.3821	0.1553
策略1+策略2	attention模型-random	0.5036	0.1961	0.5783	0.2522
	attention模型-content	0.5387	0.2082	0.6222	0.2653
	LastItem模型-random	0.3844	0.1537	0.3873	0.1702
	LastItem模型-content	0.4599	0.1857	0.5017	0.2319
融合模型		**0.6346**	**0.2521**	**0.7177**	**0.3148**

算法在 Track B 线下验证集上的结果如 Figure 1 所示：首先图中给出了四个原始模型在验证集上的效果。对比其他结果可知，在原始模型上使用策略 1 能明显提升算法在 half 的指标，并且能稍微提升 full 的指标；在原始模型上同时使用策略 1 和策略 2 也能明显提升 half 的指标，但是 full 的指标也会明显下降。最后我们提交的融合模型在线下的结果如图中的最后一行所示。

4 结论

在本技术报告中，我们介绍了我们队伍在 KDDCUP 2020 debiasing 竞赛中提出的算法。在该算法中，我们提出线性加权融合四个深度学习模型，通过两个模型框架（模型 1：**Attention 模型**，模型 2：**LastItem 模型**）分别通过两种不同的初始化方式训练得到，并设计两个效果显著的后处理策略来提升融合模型的最终效果，并取得了较好的推荐公平性。本技术报告的算法在竞赛中获得了第五名。