Lunascope Lv1
2020年02月15日 阅读 1908

关注

召回常用评估指标

背景

当前主流的推荐系统主要包括召回(recall，也称match)、排序(ranking)、重排(rerank)等三大模块。召回负责从海量候选集中根据user和item特征筛选出用户感兴趣的item。 当前召回现状如下：

1. 由于①用户兴趣多元化，用户对热门、个人偏好等均有需求，②图文视频等内容多元化等原因，召回大多为多路的形式。因此各路召回之间有交叉、互补等情况。
2. 召回的内容为排序和重排的输入，因此召回结果的好坏决定着最终推荐结果的天花板。
3. 召回位置靠前，最终效果与排序关系甚密。当前的召回大多通过hr等指标来评估，却很难实现线下线上测评数据一致，甚至出现线下测评优秀的召回，上线之后却收益甚微甚至为负的情况。

由于以上原因，召回的评估系统较难建立。为了有助于**召回算法的优化，识别模型的偏差方差，发现热门推荐和尾部个性化推荐，对齐召回和排序目标**，此文总结当前常用的算法评估方式，供召回调参使用。

概述

召回指标主要分为2部分：

1. 单路召回效果。
2. 单路召回对剩余整体的影响。

正文

1.1 Recall、Precision、F1 @N

召回最简单的3路指标，分别为Recall（用户全部点击中有多少item被召回了），Precision（召回的item中有多少被用户点击了），由于两项与N有极大关系，因此有一个调和指标F1。R(u)为给用户u的推荐列表，T(u)为用户在测试集中点击列表，#hits为用户点击的item总数。

$$Recall = \frac{\sum_u R(u) \bigcap T(u)}{T(u)} = \frac{\sum_u \#hits}{\sum_u |T|}$$

$$Precision = \frac{\sum_u R(u) \bigcap T(u)}{R(u)}$$

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision}$$

@N表示该项指标与召回个数相关，如Recall随着N的增大而增大，极端情况下N=item全集数量，此时Recall=1最大为1。而Precision随着N增大而减小，极端情况下当用户只有1个感兴趣，且该路召回第一个就是该item，Precision=1/N，那么再增大N，该值会越来越小。通常N取10，50。在实际推荐系统中可能取到100，200，多个召回队列N如何配比属于超参，且是一个巨大的搜索空间，暂没有好办法调试。

1.2 NS-Recall、NS-Precision @N

(i) We randomly select 1000 additional items unrated by user u. We may assume that most of them will not be of interest to user u.

(ii) We predict the ratings for the test item i and for the additional 1000 items.

(iii) We form a ranked list by ordering all the 1001 items according to their predicted ratings. Let p denote the rank of the test item i within this list. The best result corresponds to the case where the test item i precedes all the random items (i.e., $p = 1$).

(iv) We form a top-N recommendation list by picking the N top ranked items from the list. If $p \leq N$ we have a *hit* (i.e., the test item i is recommended to the user). Otherwise we have a *miss*. Chances of hit increase with N. When $N = 1001$ we always have a hit.

$$NS-recall = \frac{\#hits}{|T|}$$

$$NS-precision = \frac{\#hits}{N \cdot |T|} = \frac{NS\_Recall(N)}{N}$$

|T|为测试集item总个数。 这里与普通Recall，Precision有什么相似和不同之处呢？召回的计算相同之处为分子都是$R \bigcap T$，分母都是|T|，不同之处为普通Recall的R=|user| * N，即为每个用户在全部集合里匹配出N条item，但用户习惯不同，点击个数成幂律分布，大部分用户点击很少，但却对每个用户都推出了N条，此处相当于多引入了一个变量（用户的点击分布），使得不同时期的recall不可比较。NS-Recall的R=|T| * N，即为每个用户点过的项匹配出N个item，该值对每一个算法在不同时期都是稳定的。 Precision与NS-Precision的差异比较大。二者分子含义同上文Recall。分母也不同。Precision分母是召回个数即|user| * N，计算结果表示召回的item中有多少是用户感兴趣的。NS-Precision分母是整个点击+负采样集合，计算结果表示在整个候选集中，召回item中有多少是用户感兴趣的。同样避免了引入更多的变量，即用户点击的个数。

1.3 HR, ARHR @N

HR（Hit Rate）和ARHR（Average Reciprocal Hit-Rank 和位置信息相关）参考论文[2]。 #hits为测试集点击item数量，#user为用户数量，p_i为$item_i$在推荐列表经排序后的位置。ARHR度量了一个item被推荐的强烈程度，可以衡量ranking对该算法是否友好。

$$HR = \frac{\#hits}{\#users}$$

$$ARHR = \frac{1}{\#users} * \sum_{i}^{\#hits} \frac{1}{p_i}$$

1.4 MAP @K

1.5 ECS

ECS参考论文[3]。ECS（effective catalog size）的计算需要将所有item按照曝光或点击频率降序排列，i为排序后的位置，p_i为$item_i$出现的概率（即$\#item_i / \#item$），且满足$\sum_{i=1}^{N} p_i = 1$。ECS度量了item的曝光次数期望，取值在[1, N]， N=内容数， 值越大，推荐越偏向长尾，值越小，推荐越偏热门。当最流行的只有一个vid时，$p_i = 1$， ECS=1；当所有vid曝光数一样时， $p_i = 1/N$， ECS=N。

1.6 熵

p_i 为类别i在召回集中出现的概率，熵衡量了该召回的类别多样性，值越大，推荐item的列表类别越"混乱"。该值可以检验该召回是否集中在几个热门类别。

$$Entropy = \sum_i p_i * log(p_i)$$

1.7 KL散度

与上类似，p_i 为类别i在召回集中出现的概率。q_i 为类别i在另一集中出现的概率，KL散度可以衡量算法召回的结果与该集合类别分布是否一致。 当q_i 的集合为训练集时，可以衡量预测集是否和训练集类目分布一致。 当q_i 的集合为测试集时，可以衡量预测集是否和测试集类目分布一致。 在实际应用中，经常发现某些模型类算法推荐过于泛华，热门推荐突出。KL散度可以说明某个召回算法是否集中在某几个热门类。

$$KL = \sum p_i * log(\frac{p_i}{q_i})$$

1.8 长尾判断

推荐系统中，长尾效应十分明显，上述指标易受到头部数据的干扰， 且大部分召回对尾部的学习不如头部数据好，因此可以增加单独的长尾指标，参考论文[2]。 如图(图来自论文[2])，1%的item占据了20%的流量，可以去掉这1%，仅保留剩下的99%，用Recall，Precision等评估。

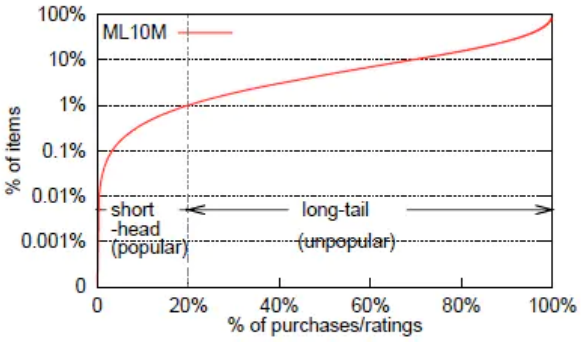

Fig. 2: Purchase/Rating Distribution in ML10M

可直接判断尾部80%的视频的推荐情况。

引用：

[1]Performance of Recommender Algorithms on Top-N Recommendation Tasks

[2]Sparse Linear Methods for Top-N Recommender Systems

[3]The Netflix Recommender System- Algorithms, Business Value, and Innovation

Lunascope Lv1 算法-应用研究 @ 腾讯
发布了 8 篇专栏 · 获得点赞 14 · 获得阅读 3,560

关注

输入评论...

相关推荐

OneChance · 1天前 · 人工智能
文本匹配模型TextMatching
2 1

OneChance · 1天前 · 人工智能
fastText简介与实践
1

OneChance · 1天前 · 人工智能
Siamese Network 孪生网络简介
1 1

行者AI · 4天前 · 测试 / 人工智能
Web UI自动化测试之元素定位
1

Jack_Cui · 3天前 · 计算机视觉 / 人工智能
StyleFlow，牛逼！
3

掘金酱 · 3天前 · 前端 / 后端 / Android / iOS / 人工智能
「掘力计划创作者训练营--引力计划」优秀学员名单公布！
9 9

txyugood · 4天前 · 机器学习 / 人工智能
手把手带你入门机器学习2——线性回归代码实现
3

又拍云 · 3天前 · WebP / 人工智能
Doge.jpg 的背后是什么，你知道么？
2 2

RAIS · 4天前 · 深度学习 / 人工智能

GoCoding · 5天前 · 图像识别 / 人工智能

MMSkeleton 快速开始，使用 WebCam 测试

2

行者AI · 7天前 · 人工智能

旋律生成算法的现状与挑战

3

算法时空 · 4天前 · C++ / 人工智能

Mac上折腾GMP

2

caiyongji · 5天前 · 机器学习 / 人工智能

机器学习(二)：理解线性回归与梯度下降并做简单预测

1 2

hongzhulei · 4天前 · 直播 / 人工智能

视频直播技术真的很难吗？手把手带你实现直播技术(二)

2 1

京东科技开发者 · 3天前 · 人工智能

JUST技术：提升基于GPS轨迹的路网推测精确度

2

Redflashing · 5天前 · 深度学习 / 人工智能

详述Deep Learning中的各种卷积（一）

4 1

Jack_Cui · 11天前 · 人工智能

用自己训练的AI玩王者荣耀是什么体验？

14 3

机器学习炼丹术 · 9天前 · 人工智能

图像分割论文 | DeepLabV1V2V3V3+四版本 | ICLR2015 CVPR2017

3

Redflashing · 5天前 · 深度学习 / 人工智能

详述Deep Learning中的各种卷积（二）

4

PureWhite · 1月前 · Go / 后端 / 人工智能

使用人工智能优化 Golang 编译器

67 9

5

凹凸实验室 · 21天前 · 人工智能 / JavaScript

凹凸技术揭秘 · Deco 智能代码 · 开启产研效率革命

41 10

人工智能AI技术 · 10天前 · 人工智能

python系列教程40

1

承志 · 14天前 · 算法 / 人工智能 / 后端

吃透论文——推荐算法不可不看的DeepFM模型

6

高德技术 · 16天前 · 人工智能

高德地理位置兴趣点现势性增强演进之路

4 1

webmote33 · 11天前 · 数据挖掘 / 人工智能

StreamSets 表达式语言篇 | 七日打卡

2

设计稿智能生成代码 · 14天前 · 人工智能

Icon 如何在设计稿生成代码中被识别？ - Imgcook 3.0 系列

9

行者AI · 13天前 · 人工智能

音频特征提取方法和工具汇总

3

机器学习炼丹术 · 18天前 · 人工智能

注意力论文解读(1) | Non-local Neural Network | CVPR2018 | 已复现

5

GoCoding · 16天前 · 人工智能

MMDetection 快速开始，训练自定义数据集

2