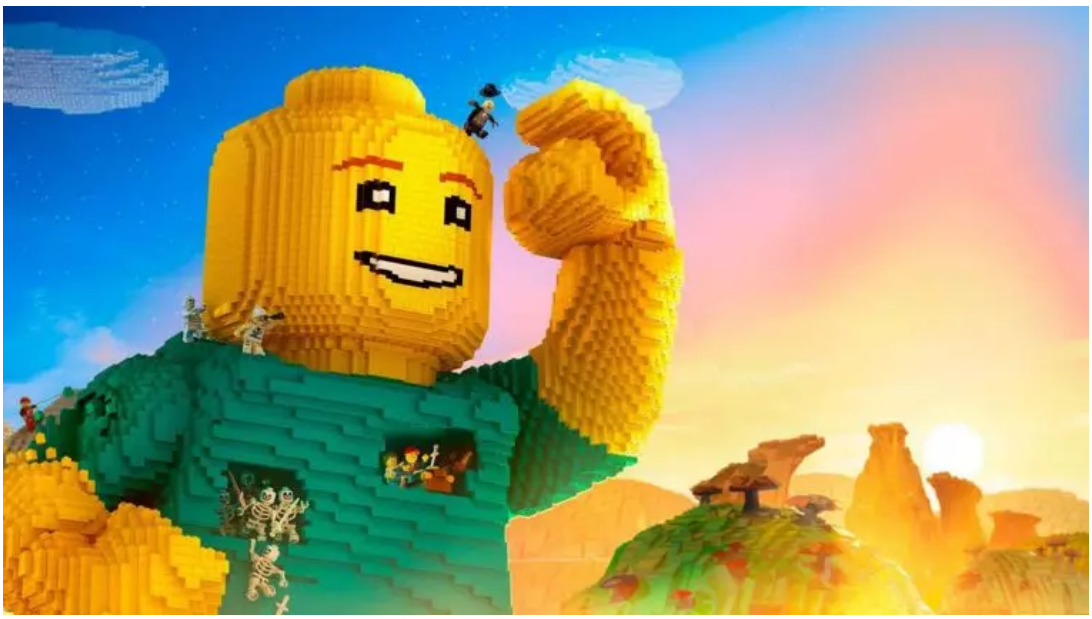
一路打怪升级，360推荐系统架构演进

51CTO学院 1月25日

> 推荐系统的核心排序算法已经从传统的 LR、GBDT 等模型进化到了 Deep&Wide、DeepFM、PNN 等若干深度模型和传统模型相结合的阶段。



如何结合各个业务数据的特点，设计合适的深度推荐算法，同时设计合理的架构保证深度学习算法的稳定运行，成为企业在推动基于深度学习的推荐系统落地的难点。

2018 年 11 月 30 日-12 月 1 日，由 51CTO 主办的 WOT 全球人工智能技术峰会在北京粤财 JW 万豪酒店隆重举行。

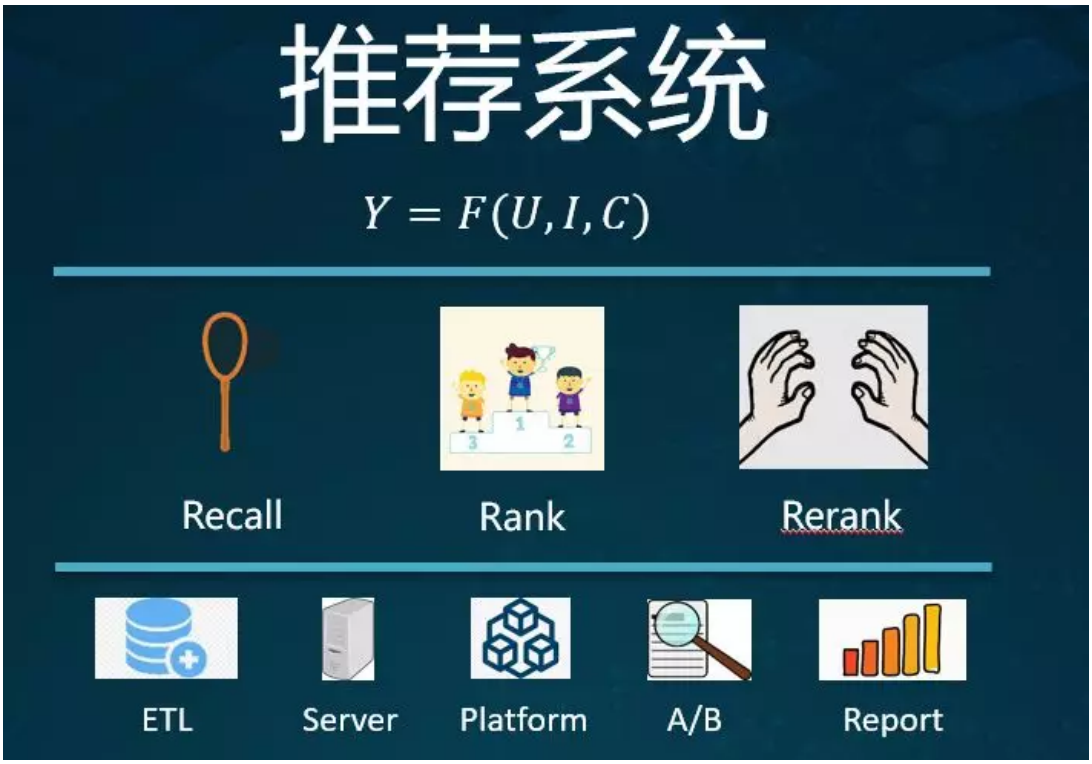
本次峰会以人工智能为主题，360 人工智能研究院的技术经理张康在推荐搜索专场与来宾分享"基于深度学习的推荐系统在 360 的应用"的主题演讲，为大家详细阐述在 360 的各种场景下，基于深度学习的推荐系统的各种应用。

本次分享分为如下两大部分：

- **推荐系统相关算法最新研究进展**
- **深度推荐系统在 360 的应用实践**

推荐系统相关算法最新研究进展

在介绍应用系统之前，首先让我们从抽象的层次上理解一下，在图像领域的相关概念。



上图是我们对推荐系统的一个分层与抽象。在顶层，我们可以理解为是一个函数。

其中 U 代表用户、I 代表需要推荐的商品、C 代表上下文、Y 则是我们需要优化的目标。

当然，不同的应用场景，Y 的取值会有一定的差异。如果我们的目标是点击率的话，那么 Y 的取值就是 0 和 1。

而如果我们要预估某个时长的话，那么 Y 的取值就变成了实数，它对应的就是某个回归问题。可见，根据不同的场景，定义好 Y 是至关重要的。

如果是从算法人员的角度出发，他们会认为定义 Y、和对 F 求解的优化是非常重要的。

而在业务方的那些做产品的人看来，U 的反馈则更为重要，如果出现用户投诉的话，那么该算法也就失败了。

另外，他们也会关注 I。由于 I 的背后实际上关联的是商家，那么同样要避免出现用户对于 I 的抱怨。可见，不同角色对于此公式的关注点是不相同的。

在上面抽象图的中间，我们一般会把顶层简单的数学公式拆分成三个不同的算法模块：

- **召回（Recall）**
- **排序（Rank）**
- **策略（或称重排序 Rerank）**

目前市面上的一些工业领域和学术界的论文，大部分会重点研究和讨论 Rank，毕竟 Rank 是非常重要的。

而对于那些针对 Recall 和 Rerank 的技术而言，由于它们并不适合被抽象成为一个统一的理论架构，因此相关的论文也不多。后面我们会重点讨论有关召回部分的内容。

经历了上面两个抽象层次，图中的底层就需要让推荐系统服务于“线上”了。

它由五大关键部分所组成：

- **ETL 对数据的清洗。**不同于那些已经准备好了数据集的传统竞赛，我们面临的是在真实的线上场景中所产生的日志数据，它们不但“脏”，而且体量非常大。
  因此，我们需要有一个对应的数据清洗场景，以缓解系统的处置压力。
- **Server 模块。**针对各种排序和召回的模型场景，我们需要提供实时的服务。
  因此服务端不但需要具有高性能的计算能力，同时也需要我们的架构能够应对大规模的深度学习与计算。如有必要，还可能会用到 GPU 等硬件。
- **Platform。**这里主要是指深度学习或者机器学习的训练平台。在各种算法上线之后，随着在线学习的推进，其模型不可能一成不变。
  有的它们需要被“日更”，甚至是以分钟为单位进行更新。因此我们需要有一个良好的深度学习平台提供支持。
- **测试。**推荐系统在上线之后，需要被不断的迭代与优化，因此我们需要通过测试来查看效果。
  在系统的起步阶段，用户数量迅速上升，而实验的整体数量则不多，因此我们很容易通过对百万级用户的切分，来开展与流量相关的实验。
  但是随着业务的发展，用户数量不再呈爆发式增长，而我们每天又需要进行成百上千次实验，所以我们需要选用 A/B 测试的实验平台，以方便算法人员加速迭代的进程。
- **报表。**之前在与业务方合作时，我们发现：他们几乎每个人都在通过自行编写简单脚本的方式获取所需的报表，因此其工作的重复度相当高。
  然而，由于许多报表的计算都是简单算子的累加，如果我们拥有一个简单且统一的平台，就能够帮助大家获取常用的指标，进而加速整个系统的迭代。

从深度推荐系统的发展来看，最早出现的是传统 LR（线性回归）的机器学习。

之后，随着特征交叉需求的增多，出现了非线性回归和使用 FM 来实现二阶特征交叉。

近年来，随着深度学习在图像领域的广泛应用，如今大家也将它们引入到了推荐系统之中。

不过，相对于图像领域动辄一两百层的神经网络深度而言，推荐系统的深度只有四到六层。

如今各家“大厂”都能够提供诸如 FNN、DFM、以及 Google Wide&Deep 之类的算法，我们很难断言哪种模型更好。

因此大家在进行模型选取的时候，应当注意自己的系统偏好，保持与业务的强关联性。

上图借用的是阿里的 DIEN（Deep Interest Evolution Network），他们和我们现在的工作有着相似之处，特别是图中虚线框里的两个创新点。我会在后面进行着重分析。

平时，我们在阅读各种论文时，很容易产生一种错觉：那些作者为了突出自己的贡献，经常会着重描写模型上的创意，体现并抽象其核心的思想，让大家认为该部分的占比能达到 80%。

而实验部分都只是一些标准的数据集，经常一笔带过。实际上，他们在模型侧的工作量一般只有 20%，其他的 80% 则花在了做实验上。

具体说来，我们可以开展如下各种实验：

- 在拿到原始的线上日志数据之后，我们需要对它们做分析、统计和处理，以方便我们后期进行模型训练。
- 特征设计、交叉和服务，包括：是否要将特征值离散化，是否要放大时间窗口等。
- 模型选择，我们需要全面考虑计算的开销，甚至要考虑自己的平台上是否有 GPU，倘若没有，则不应选择过于复杂的模型。同时，我们也要根据核心算法，制定召回和仲裁策略。
- 最后是真正传统意义上的实验，包括离线评测、线上 A/B 和实验分析等，这些都会与前面的四项有所关联。

深度推荐系统在 360 的应用实践

下面，我们来讨论一下深度推荐系统在 360 中几个场景应用的落地。如图的三款应用在技术上呈由浅入深、由简单到复杂的递进关系。

场景一：App 推荐

360 手机助手，类似于应用市场。图中红框里显示的都是系统推荐的内容，包括：“今日热点”和“大家都在玩”等，这些都是与用户的个性化喜好相关的。我们直接从用户的下载和转化率，来判定系统推荐的效果。

套用前面提到的三个层次：

- **抽象定义层。**由于我们做的是一个离线系统，因此我们对 C 考虑并不多，主要将重点放在了 U 和 I 上。该场景下的 U 属于亿级，而 I 只有万级，毕竟我们只需要在万级的 App 库里做推荐。

- **算法策略层。**由于 I 的量级比较小，我们当时就“简单粗暴”地直接在这个万级用户库里做了用户匹配和推荐，而并没有执行召回策略。我们首先通过简单的算法做出了一个初版模型。为了查看效果，我们通过迭代、优化和序列预测，来预测用户下一次会安装哪一种 App。而 Rank 和 Rerank 部分的逻辑，由于主要涉及到各种线上的业务逻辑，因此是由业务方来完成的。
- **架构层。**它对应的是我们的天级离线中心，在特征上并不复杂，仅通过 Embedding 特征的运用就完成了。

上图是我们对于模型的逻辑进化路径。上图的左侧与 Google Wide&Deep 的思路是很相似的，只不过我们用的特征非常少。

上方蓝色部分是用户 App 序列的 ID 列表，我们使每个 App 都有一个特征向量，然后做若干层的 MLP，最后做分类，从而预测某个用户下一步可能安装的 App。

后来，随着与业务人员的交流，我们在模型中加入了用户的浏览历史、搜索历史、关键词特征、和 App 特征等元素，不过其本质上也都是行为的序列。

考虑到简单的 MLP 已不太适合，因此我们就尝试采用了 RNN 的方法进行数据建模，于是就有了如下图进化后的网络：

虽然该图的上层分类与前面类似，但是下面的 browse 却值得大家注意：当你把用户的搜索行为加到自己的特征里时，请注意调整时间序列，不然很容易会导致该搜索词变成了一个搜索系统。

即：你的本意是通过用户的搜索词，来反应他的兴趣，并预测他要下载的 App。

但是，如果不做处理的话，就可能导致你的系统只是在模拟某个线上的搜索。

上面这张图是将两个模型的不同结果进行了比较。它们分别对每个 App 的 Embedding 向量进行了可视化。我们将 App 的 Embedding 向量取出，并投影到二维图上。

我们根据 App 原来所隶属的大类，来判断经过模型学习后，如果能将大类拉得比较散的话，那么效果就会更好些。

可见，在左边 MLP 的中间，各种类型的交叠比较严重。而在我们换成了 RNN 模型之后，它们就显得分散了许多。我们籍此来直观地认为改进后的模型更好。

根据上面将 RNN 的效果与 MLP 的效果所进行的数据比较，我们可以看出：在 Top1 上的差异并不太大，仅提升了 1 个点。

当然，这可能对于线上系统的提升会反应得显著一些，但是对于我们的离线系统，提升反应就没那么明显了。这便是我们整个团队在推荐领域的第一次试水。

场景二：360 搜索和导航

有了前面 App 推荐的经验积累，我们尝试的第二个项目是“常搜词”。即：360 页面有自己的搜索和导航，而在搜索框的下方，我们会放置六到七个用户经常搜索、或者是时常用到的关键词。

如此，用户在打开导航时就能看到自己想看的东西，这不但减少了他们在操作上的复杂度，还提升了用户在使用上的满意度。

在该场景下，由于我们的导航每天都有好几亿次的 PV（Page View），因此使用该功能的用户有着千万级的体量，即 U 为千万级。

同时，为了定义“常搜词”，我们对 query（查询）进行了清洗，过滤掉了一些非常“长尾”的 query，只留下一些有实体字的、较为明确的 query。因此，此处的 I 就根据 query 的数量被设定为了百万级。

由于该业务本身已存在了较长时间，因此在算法策略上也已实现一些较为完善的召回（Recall），例如：

- **搜索词**：是根据用户的搜索行为，召回一些新的关键词。
- **URL**：是根据用户浏览器里面的浏览记录，再输入一些关键词。
- **广告**：是通过满足用户的兴趣匹配，精准地投放商业化的广告词。
- **百科**：是判断用户可能感兴趣的知识，投放相应的百科词汇。

另外在 Rank 方面，我们着手将线上系统从以前简单的 LR+FTRL，升级并优化成为了 DNN 模型。

在架构上，由于该导航系统的特点是用户使用频次不高，而我们的推荐可能每天只会被早晚各用一次，因此我们的目标是天级离线更新。

其对应的特征部分，则主要是基于用户的搜索日志和浏览日志，来进行的一些统计特征的设定。

让我们来看看其中几个权重比较正向的特征，例如：

- **week_freq_num 特征**，描述了用户常搜词汇的概率分布，即：如果某个用户在过去一段时间窗口内成均匀的访问趋势，则说明这是经常性的需求。该特征比较重要。

- **global_query_uv 特征**，此类全网特征能够在一定程度上反映商品的质量。我们籍此在原有特征的基础上扩展到成千上万个。

在我们将模型调整成为 DNN 之后，系统的 AUC 从 7 提到 8。之后我们还增加了一些其他能够提升 AUC 的特征。

为了细粒度地分析 DNN 模型的召回策略在上线之后所产生的影响，我们拿它与传统的 LR+FTRL 模型进行了对比。

途中两处 CTR 较高，它们都与用户的搜索和浏览行为相关，并且能够真正反映他们的需求。

由于此类搜索、浏览和 MLP 的点击率本身就比较高，因此我们需要在此基础上有一个量的提升。

另外在上图中，虽然与个性化相关的广告部分略有提升，但是 ys_mid_hist 是有所下跌的。

通过分析，我们发现该模型当时在排序时，倾向于把策略后排，其导致的结果是：用户在界面上默认只能看到六个推荐词，而实际上其旁边有一个往下点的箭头，在被点击之后才会出现很多的词汇。

因此，如果某些推荐词汇被排到了第六名之后的话，那么它们被点击的可能性就非常小。这也说明了默认六个词的重要性。

同时，这也提示了我们：在做数据时，需要重点关注那些用户通过点击下拉按钮去访问的词汇，那些数据才更能够说明用户对某些词汇的强烈需要。总的说来，我们协助业务侧将 2017 年 Q1 的收入提升了 5%。

场景三：快视频

基于前面两个项目所积累的经验，我们全情地投入了第三个项目—快视频。其主要的业务形式是个性化的消息推送，与大家手机上的通知栏推送相类似。

但是，略有不同的是：我们并非每日都定时推送，而是会根据用户的反馈偏好，去调整推荐的数量、频率和时间。

同时，这对于将一些尚未养成每日到我们的 App 里观看视频的用户来说，能够起到一定“拉活”的作用。

在此，我们仍然套用前面的三个层次。在顶层，由于该 App 上线之后，视频库的规模和用户的数量都增长得非常迅速，因此 U 和 I 都具有千万级。

在召回策略上，图中列出了五种重要的召回。前三项分别是：语义、视觉和行为召回。我们统一地对它们进行了重点优化。

而在距离用户更近的 Rank 部分，我们通过收敛，采用了 LSTM + Attention 的模型。

前面介绍过的两个项目都仅仅是以天为单位进行更新，因此对算法人员的压力并不大，只需采用定时任务便可。

而在这个场景中，由于它是一个对用户进行实时在线推荐的系统，如果采用“天级更新”频率的话，会给用户带来较差的体验。

因此无论是对召回的数据流、模型的更新、还是在线的服务，我们都认真进行了架构设计。

我们来看看刚才提到的三个召回策略：行为、语义和视觉。在此，我们将它们统一到了 Embedding 框架之中，以便从视频的不同角度进行描述和特征表示，而不必关心后续有关设计域值、队列长度等问题。

通过执行统一的流程，我们能够更直观的体现出各种视频在特征领域下召回的根据。

例如图中黑体字是某个视频的 title（标题），明显，它是有关传授烹饪知识的召回：

- **就行为模型而言，**它通过对用户行为序列的协同与分割，为每个视频分配各种特征向量。其特点是：会产生一定的扩散效果。即，你会在上图中发现，该召回的前两个结果已经不再关注烹饪，而是扩散到了其他领域。只有第三个召回才是真正有关于烹饪的。
- **语义模型，**重点关注的是视频标题的语义表示，通过语法和语义的相似性，此类召回能够抓住标题中关键性的动词与名词，然后进行相应的交叉和扩散等操作。
- **视觉模型，**是通过抽取视频的特征，来实现召回。它与标题的语义关联性不强，就算标题与烹饪无关，只要在视频中大量出现食物的图像，便可以表示并召回。

由上可见，我们通过不同的 Embedding，就可以实现多样的召回策略。

下面我们重点介绍语义模型。我们所得到的输入数据是公司过往生成的查询、与搜索数据，或是其他的业务线积累的标签数据，例如由网页标题所抽象出来的简单标签。

这些标签既有自动生成的，也有通过人工修正和 rebind（重新连接）产生的。它们都具有一定的实际价值。

为了给每个 title 产生相应的表征，我们构建了一项任务：首先是对每个句子进行分词，以得到相应的字/词向量，然后通过 CNN 网络进行处理，最后对目标予以多分类、多标签，从而预测该标签准确性。

对于上述任务，我们除了尝试过 CNN 模型之外，还在不改变 task 的情况下试用了双向的 LSTM，即 bi-LSTM。接着，我们需要对该语义模型进行评测。

在此业务场景下，我们人工标出了几千个具有良好多样性的数据，并将其作为离线评测的指标，以此来检查通过语义模型召回标注数据的结果相关性。上图左侧的表格就是我们测试的结果。

我们在此比较了几种常见的语义建模的模型、及其对应的优化。表格的中间列是 Title2Tags，即通过标题来预测对应的标签。

至于右边列：Title2UnionTags，我们对应地发现到：如果只有单个标题，则语义往往会被遗漏。

如上图右侧的例子所示：无论是人工写的标题，还是自动生成的，如果我们将多个标签放到一块儿，句子中的语义则会更准确地被覆盖到。

另外我们后续对数据的清洗，也能够提升原句子的标签数量，进而大幅提升模型的整体效果。

上图是我们开篇引用过的阿里 DIEN，在其虚线框中有两个创新点，分别是 AUGRU（即 Attention 与 GRU 的合并）和辅助的 loss。

虽然 loss 对模型的提升效果非常明显，可惜我们在模型设计时，过于关注 Attention 的 layer 和下沉到用户序列上的特征，因此我们将来尚有改进的空间。

上图便是我们当时对不同排序模型的评测结果。

可见我们在系统架构的设计上，既有诸如到台服务、日志搜集、和报表等离线部分，又有实时的在线更新。

同时，我们也与业务方的在线系统保持着实时性的交互，以便他们按需获取推送的数据。另外在最上层，我们为业务方也提供了多种安全策略和分桶策略。

上图便是我们从 2017 年 11 月 1 日到 2018 年 1 月 31 日的 CTR 曲线。

虽然有某些“坑点”所导致的曲线波动，但是整体趋势还是向上的，特别是在我们更换了 Attention 模型之后，提升的幅度能够达到 10%，进而有效地帮助业务方实现了“拉活”和保持更多的活跃用户。

综上所述，我们基于深度学习的推荐系统就是根据上述三个层次，一步一步地通过迭代和优化发展起来的。

作者：张康

介绍：奇虎 360 技术经理，毕业于清华大学计算机系，获博士学位，2015 年毕业后加入奇虎 360，2016 年加入 360 人工智能研究院。主要参与深度学习算法在公司实际业务的落地，先后参与了 App 推荐、搜索词推荐、导航购物推荐、短视频个性化推送等项目。

编辑：陶家龙、孙淑娟

来源：有投稿、寻求报道意向技术人请联络 editor@51cto.com

阅读原文