答疑 | 线上服务篇留言问题详解

你好，我是王喆。

今天是专栏的第二次答疑加餐时间，第一次答疑我已经对基础篇和特征工程篇中常见的问题进行了解答，所以这节课我们重点来看看线上服务篇中很有代表性的课后思考题和留言问题，我会对它们进行一些补充回答，希望对你有帮助。

关于项目的开源精神

在开始回答问题之前，我想先跟你聊一聊我们 SparrowRecsys 项目的开源精神。在课程一开始我就说过，SparrowRecsys 这个项目是我们的一个种子项目，它肯定不完美，但我希望它是一个工业级推荐系统的雏形。在学习的过程中，我希望能够跟你一起完善它，让它的羽翼逐渐丰满起来。

让我很高兴的是，已经有不少同学投身到改进 SparrowRecsys 的队伍中来，比如 GitHub ID 叫dxzmpk的同学添加了Node2vec 模型的代码，还有 GitHub ID 叫 jason-wang1 的同学添加了多路召回多线程版本的代码，还有更多的同学修改了项目中的 Bug，优化了一些实现，感谢你们的投入！

我是开源精神的坚定拥护者，我也相信在我们的共同努力下，SparrowRecsys 未来能够发展成为在业界有影响力的开源项目。所以在这里我呼吁同学们能够多参与进来，多提 Pull Request，让我们共同成为项目的第一批原作者。

好，下面我们进入问题解答的环节。

《03 | 深度学习基础：你打牢深度学习知识的地基了吗？》

思考题 1：哪些因素影响着深度学习网络的结构？深度学习模型是越深越好吗？

这两个问题我们分开来看，先看看影响深度学习网络结构的因素。在业界的应用中，影响深度学习网络结构的因素非常多。不过，我认为可以总结出二类最主要的因素。

第一类：业务场景中用户行为的特点。很多模型结构的实现是为了模拟用户行为的特点，比如注意力机制的引入是用来模拟用户的注意力行为特点，序列模型是用来模拟用户兴趣变迁，特征交叉层是为了让用户和物品的相关特征进行交叉等等。

第二类：数据规模、算力的制约。这一点“一天”同学回答得非常有价值，在实际的业界应用中，数据规模大起来之后，我们往往不能够随意选择复杂模型，而是要在数据规模，平台算力的制约下，尽量选择效果最优的模型结构。

我们再来看第二个问题，深度学习模型是越深越好吗？

这个答案是否定的。深度学习模型变深之后，并不总是能够提高模型效果，有时候深度的增加对模型效果的贡献是微乎其微的。而且模型复杂之后的负面影响也非常多，比如训练时间加长，收敛所需数据和训练轮数增加，模型不一定稳定收敛，模型过拟合的风险增加等等。所以在模型深度的选择上，我们要在尽量保证效果的前提下，选择结构较简单的方案。

借助这道思考题，我希望能帮助你更好地理解深度学习的特点，以及实际应用中的一些经验。

《09 | 线上服务：如何在线上提供高并发的推荐服务？》

思考题 2：在一个高并发的推荐服务集群中，负载均衡服务器的作用至关重要，如果你是负载均衡服务器的策略设计师，你会怎么实现这个“工头”的调度策略，让它能够公平又高效地完成调度任务呢？（比如是按每个节点的能力分配？还是按照请求本身的什么特点来分配？如何知道什么时候应该扩展节点，什么时候应该关闭节点？）

负载均衡的策略其实有多种选择，比如“smjccj”同学的回答就很专业，他说可以进行源地址哈希，或根据服务器计算能力加权随机分配。这是一个很好的答案，这里我再补充一下。通常来说，常用的负载均衡的策略有三种，分别是轮询调度、哈希调度和一致性哈希调度。我们一一来看。

轮询调度就是以轮询的方式依次把请求调度到不同的服务器。在服务器的算力等硬件配置不同的时候，我们还可以为每个服务器设定权重，按权重比例为能力强的服务器分配更多的请求。

而哈希调度指的是通过某个哈希函数把 key 分配给某个桶，这里 key 可以是请求中的用户 ID，物品 ID 等 ID 型信息，桶的总数就是服务器的总数。这样一来，我们就可以把某个用户的请求分配给某个服务器处理。这么做的好处是可以让一个 key 落在固定的服务器节点上，有利于节约服务器内部缓存的使用。

哈希方式的缺点在于无法高效处理故障点，一旦某个点有故障需要减少桶的数量，或者在 QPS 增大时需要增加服务器，整个分配过程就被完全打乱。因此，一致性哈希调度就是更好的解决方案，简单来说就是使用哈希环来解决计算节点的增加和减少的问题，具体的实现我推荐你参考《一致性哈希算法的理解与实践》这篇文章。

留言问题 1：在一个成熟的工业级推荐系统中，每个用户请求的时间、地点、context 都不一样，缓存策略是怎么工作的，才能把这些数据大部分都缓存起来？

雪焰

写于 2020年10月24日

王老师好，请问对于：“在一个成熟的工业级推荐系统中，合理的缓存策略甚至能够阻挡掉 90% 以上的推荐请求，大大减小推荐服务器的计算压力”，我理解每个用户的请求，在不同的时间，地点，context 下是不一样的，这样千差万别的请求，数量级应该很大，不知道是如何把大部分都缓存起来的呢？谢谢

深度学习推荐系统实战

09｜线上服务：如何在线上提供高并发

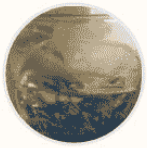
识别二维码打开原文
「极客时间」 App

这里，同学们一定要理解缓存的意义。如果请求中的变量每次都不一样，那我们确实就没有必要进行缓存了，因为每次返回的结果都是不同的。但真实情况往往不是这样，我们其实可以在具体的业务场景中挖掘出巨大的优化空间。

比如，电商网站存在着大量没有购买记录的新用户，我们其实可以根据这些新用户有限的特征把他们分成少量的几个类别，对一个类别内的用户展示同样的推荐结果。这样，我们就没必要每次都请求复杂的推荐模型了。

再比如，同一个用户有可能多次请求同一个页面，如果推荐系统对这些操作进行了缓存，就不用对每次重复的请求重复计算推荐结果了，在处理完首次请求之后，面对之后的重复请求，推荐系统直接返回缓存结果就可以了。当然，推荐系统具体能存储多少用户缓存，也取决于硬件配置，也取决于缓存的过期时间，这些都需要我们灵活进行配置。

留言问题 2：推荐系统中的冷启动策略指的是什么？



那时刻

写于 2020年10月29日

请问老师，按照一些规则预先缓存好几类新用户的推荐列表，等遇到新用户的时候就直接返回，这算是冷启动策略么？

深度学习推荐系统实战

09｜线上服务：如何在线上提供高并发

识别二维码打开原文
「极客时间」App

冷启动是推荐系统一定要考虑的问题。它是指推荐系统在没有可用信息，或者可用信息很少的情形下怎么做推荐的问题，冷启动可以分为用户冷启动和物品冷启动两类。

用户冷启动是指用户没有可用的行为历史情况下的推荐问题。一般来说，我们需要清楚在没有推荐历史的情况下，还有什么用户特征可以使用，比如注册时的信息，访问 APP 时可以获得的地点、时间信息等等，根据这些有限的信息，我们可以为用户做一个聚类，为每类冷启动用户返回合适的推荐列表。当然，我们也可以利用可用的冷启动特征，来构建一个较简单的冷启动推荐模型，去解决冷启动问题。

对于物品冷启动来说，主要处理的是新加入系统的物品，它们没有跟用户的交互信息。所以，针对物品冷启动，我们除了用类似用户冷启动的方式解决它以外，还可以通过物品分类等信息找到一些相似物品，如果这些相似物品已经具有了预训练的 Embedding，我们也可以采用相似物品 Embedding 平均的方式，来快速确定冷启动物品的 Embedding，让它们通过 Embedding 的方式参与推荐过程。

《11 | 召回层：如何快速又准确地筛选掉不相关物品？》

留言问题 3：用户的多兴趣标签怎么与物品的标签进行最优匹配？当物品的标签有多层时，如何利用上一层的标签？

Geek_e0d66a

写于 2020年10月29日

请问老师，如果基于兴趣标签做召回，同一个物品，有多个标签，而用户也计算了出了多个兴趣标签，那么怎么做用户的多兴趣标签与物品的最优匹配呢？还有物品的标签有多层，那么怎么利用上一层的标签呢？

深度学习推荐系统实战

11 | 召回层：如何快速又准确地筛选指

识别二维码打开原文
「极客时间」 App

这个问题最简单的做法，就是把用户的兴趣标签和物品对应的标签都转换成 Multi-hot 向量，然后，我们就可以计算出用户和物品的相似度了。

除此之外，我们也可以进一步计算每个兴趣标签的TF-IDF 值，为标签分配权重后，再把它们转换成 Multi-hot 向量，这样我们也可以计算出用户和物品的相似度。

如果标签有多层，我们也可以把多层标签全部放到 Multi-hot 向量中，再把高层标签的权重适当降低，这也是可行的思路之一。

留言问题 4：在电商领域下，如何解决 EGES 训练非常慢的问题？

萬里長空

写于 2020年10月29日

老师, 关于 EGES 的训练, 试了下, 由于电商领域商品维度非常大, 即使 hash 后也很大, 这导致训练非常慢, 这个一般怎么解决啊?

深度学习推荐系统实战

11｜召回层：如何快速又准确地筛选掉

识别二维码打开原文
「极客时间」 App

这是一个非常好的业界实践问题。这里，我先给同学们解释一下，什么是 EGES。EGES 指的是阿里提出的一种 Graph Embedidng 方法，全称是 Enhanced Graph Embedding with Side Information，补充信息增强图 Embedding。它是一种融合了经典的 Deep Walk Graph Embedding 结果和其他特征的 Embedding 方法。

针对 EGES 的训练比较慢的问题，我这里有两条建议可供同学们参考。

第一条是我们可以把商品 Embedding 进行预训练，再跟其他 side information 特征一起输入 EGES，不用直接在 EGES 中加 Embedding 层进行 End2End 训练。

第二条是我们可以把商品进行聚类后再输入 EGES 网络，比如非常类似的商品，可以用一个商品聚类 id 替代，当作一个商品来处理。事实上，这种方法往往可以大幅减少商品数量的量级，AirBnb 就曾经非常成功地应用了该方法，用一些特征的组合来代替一类商品或用户，不仅大幅加快训练速度，而且推荐效果也没有受到影响。

《12｜局部敏感哈希：如何在常数时间内搜索 Embedding 最近邻？》

留言问题 5：在用 Item2vec 等方法生成物品 Embedding 后，用户的 Embedding 是怎么生成的呢？ 物品和用户在同一个向量空间，这是怎么保证的呢？

浣熊当家

写于 2020年10月31日

请问老师关于这句话“在训练物品和用户的 Embedding 向量时，如果二者的 Embedding 在同一个向量空间内”， 我们在之前 6-7 节 embedding 的中，讲了怎么把物品序列信息转化为 embedding， 想知道，用户的 embedding 是怎么生成的呢？ 然后，物品和用户在同一个向量空间，这个是怎么得到的呢？

深度学习推荐系统实战

12｜局部敏感哈希：如何在常数时间内

识别二维码打开原文
「极客时间」 App

在咱们的项目里，用户 Embedding 的生成方法是很直观的，就是对用户评论过的高分电影 Embedding 取平均值得到的。这相当于说，用户 Embedding 是在物品 Embedding 向量空间中进行运算得到的，那它们肯定是在一个向量空间内，我们也就可以使用相似度计算来求取相似度。

因此，只要是利用用户历史的 Item Embedding 生成的用户 Embedding，都是在一个向量空间内，这些生成方式包括 average pooling、sum pooling、attention 等等。

但是如果用户 Embedding 和物品 Embedding 是分别独立生成的，或者说是通过一个模型中没有直接关系的两个 Embedidng 层生成的，那么它们就不在一个向量空间内了。注意啦，这个时候，我们不能直接求用户和物品之间的相似度，只能求用户 - 用户的相似度，和物品 - 物品的相似度。

留言问题 6：“在局部敏感哈希的函数中，b 是 0 到 w 间的一个均匀分布随机变量，是为了避免分桶边界固化”。这是什么意思呢？是说可以通过调整 b 来形成另外一个个 Hash 函数？

Dikiwi

写于 2020年11月01日

b 是 0 到 w 间的一个均匀分布随机变量，避免分桶边界固化。这是什么意思呢？是说可以通过调整 b 来形成另外一个一个 hash 函数？

深度学习推荐系统实战

12｜局部敏感哈希：如何在常数时间内

识别二维码打开原文
「极客时间」App

首先，我要说这个局部敏感哈希相关的问题非常好，推荐其他同学也关注一下。

说回到这个问题，如果我们总是固定分桶的边界，很容易让边界两边非常接近的点被分到两个桶里，这是我们不想看到的。所以，这里我们就可以通过随机调整 b 的方式，来生成多个 Hash 函数，在进行多个 Hash 函数的分桶之后，再采用或的方式对分桶结果进行组合查找最近邻向量，就可以一定程度避免这些边界点的问题。

好了，这节课的答疑就到这里，非常感谢同学们的积极提问和思考。希望在接下来的课程里，你也能多多参与进来，与我一起完善 SparrowRecsys 项目的代码，共同进步！