推荐系统中的召回阶段深入探讨

整个推荐系统的实现流程可以大致划分为三个阶段（注意，**我们下面讨论的推荐系统是做线上推荐的，区别于离线做推荐的情况**）：

召回阶段，排序阶段，重排阶段。（**从实现上的复杂性以及重要性来说，召回阶段 > 重排阶段 > 排序阶段**）

每个阶段是有所区别的，我们今天只关注召回阶段，对于召回阶段我们庖丁解牛，罗列出来重点关注的问题如下（**下面的问题基本上是按照顺序来思考的，其中会涉及到长尾用户/冷启动用户和其他用户，首页推荐和详情页推荐两种场景，召回策略等这些东东的组合情况的讨论**）：

**针对当前的某个具体业务场景，是否需要召回，排序，重排三阶段**？

**常用的召回策略有哪些？适用于我们的业务场景的召回策略有哪些**？

**召回的范式有哪些？针对我们的业务场景的召回范式的选择是什么**？

**召回阶段应该使用哪些种类的特征**？

**每个召回策略返回的topK item物品集合的依据是什么**？

**采用离线的召回结果还是做实时召回**？

**所有的用户/user都走这些召回策略吗**？

**如何评价某个召回策略的好坏**？

**需要为了将就复杂耗时的排序模型来削弱召回阶段吗？**

下面我们针对每个问题来展开详细讨论：

1. **针对当前的某个具体业务场景，是否需要召回，排序，重排三阶段**？

首先，我们要**从本质上**看清楚这三个阶段：

召回：**在资源受限的情况下，尽可能找到当前用户可能感兴趣的尽量大的item候选集**；

排序：**从多种不同召回策略得到的item候选集需要一个统一的打分方法**；

重排：**需要对排序阶段输出的候选集进行人工运营干预来生成最后的推荐列表**；

另外，**从对输入的item数量级的要求上，三个阶段也差别很大**：

对于召回阶段，或者在线上的时候直接读取保存在NoSQL中的离线预先计算好的结果，或者使用高性能的线上item向量相似度检索库来获取Topk的结果，所以把整个物品池item pool作为输入是可以的也正是我们所想要的，而物品池的物品可能有几百万甚至上千万；排序阶段一般使用传统机器学习模型或者深度学习模型，为了让模型在有限的时间内能处理完，需要排序阶段的输入item数量也就是召回的候选集总数量不能很大（一般是千级别）；重排阶段最后会生成推荐列表，推荐列表一般不会太长，所以这里的输入item数量也就是排序后取的topK的K数量一般是百级别（这里的输入item数量指的是从排序的结果集中选择多少，重排阶段的输入源除了排序后的结果集，在首页推荐场景下，一般还有优选物品集以及需要探索的那些长尾物品集和冷启动物品集）。

**三连问**：

1. 如果我的整个item集合也就是100个左右，那么需要召回阶段吗？
2. 如果就一路召回，那么还需要排序阶段吗？（这个要分情况讨论，参见下面的讨论）
3. 重排阶段一定需要吗？

**在推荐系统中，常见的两个场景是：首页推荐和详情页推荐**。

现代化的推荐系统，首页推荐是一定会考虑个性化的，详情页推荐是否考虑个性化可能每个公司的实践都不一样。

对于详情页推荐，如果不考虑个性化，那么利用item2item关联推荐的思路来召回，比如：基于item表示向量的相似度的召回，基于item关联规则的召回，基于item表示向量聚类的召回等等；如果考虑个性化即考虑用户画像（包括用户的行为）的话，把用户画像作为特征建模到排序阶段**，这样得到的推荐结果可能对用户的体验更好**。

因此，**我们在本文接下来的讨论中都假设详情页推荐需要考虑个性化**。**不管在详情页推荐中是否考虑个性化，召回阶段两者基本是类似的**。

**首页推荐的时候没有办法获得用户当前的意图，一般会使用尽可能多的召回策略：**

这里假设的是在首页推荐的时候只用个性化推荐一种方法，而所有的召回策略作为个性化推荐的召回阶段；它区别与首页使用多种推荐方法，比如Netflix的首页每一行对应一个推荐方法（比如一行是独播推荐，一行是热门推荐，一行是新品推荐，一行是个性化推荐（这里的个性化推荐的召回策略就比上面那种方法的召回策略少一些了）等等）。**本文中我们假设首页推荐只有一种推荐方法即个性化推荐。**

**详情页推荐的时候则是用户有了明显的意图（至少可以假设他对当前详情页item感兴趣），因此一般来说详情页推荐时不会有太多召回策略。**

因此，**对于首页推荐的话，一般会有完整的召回，排序，重排三个阶段：**

对于长尾用户和冷启动的用户，可能走单独的链路效果更好。

对于长尾的用户和冷启动的用户经过召回阶段后，可能不走排序阶段会更好，在重排阶段对召回的结果进行编排以及做探索。

**而对于详情页推荐或者类似的场景，分情况讨论**：

如果只有一路召回策略比如基于item表示向量的相似度的召回，并且不考虑个性化，这个情况下可以不需要排序阶段了；

如果只有一路召回但是要考虑个性化即想要考虑更多的用户画像特征，则需要一个复杂的算法/模型引入这些特征以及其他特征来对单路召回的topK的排序结果重新打分排序，这个时候就需要排序阶段了；

如果使用了多路召回：

对于长尾用户和冷启动用户来说，可能不走排序阶段会更好，使用重排阶段对多路召回的结果进行编排。

对于非长尾和冷启动的用户，走排序阶段。

对于冷启动的物品，可能通过某种召回策略（比如基于物品的显式画像表示向量召回）召回了，可能也不适合走排序阶段，使用重排阶段在某些固定位置给召回的冷启动物品更多的一些曝光机会。

**在详情页推荐的场景下，一般不会在重排阶段使用探索方式**。

小结：**不是每个推荐系统的业务场景都需要召回，排序，重排三个阶段！！！！**需要根据具体业务场景来判断是否需要召回，排序，重排的任意一个阶段。

1. **常用的召回策略有哪些？适用于我们的业务场景的召回策略又有哪些？**

**首页推荐和详情页推荐的召回策略可能不太一样**。对于“首页推荐”，什么召回策略都可以堆砌；而对于“详情页推荐”，最通用的召回策略可能就是基于item表示向量的相似度的召回，因为对于这个场景，用户的意图很明显，需要**以当前详情页的物品为中心做相关的召回**，所以很多适用于首页推荐的召回策略可能不太适合用在详情页推荐的召回阶段。

召回策略理论上来说可以使用任何方法，包括人工规则，统计方法，数据挖掘方法以及机器学习方法（包括深度学习方法）等等。

**对于首页推荐的场景，虽然说不同的业务场景，召回策略可能会有所不同，但是通用的召回策略大致可以分为如下4大类**：

**基于热度/流行度的召回，基于协同过滤的召回，基于物品画像的召回，基于用户画像的召回**。

对于一些基于地理位置的应用的推荐业务比如外卖服务或者打车服务，最终推荐的结果需要考虑当前用户的地理位置（**因为这样才合理，需要回归现实条件**）。对于这样的应用，它的召回策略与通用的召回策略有一些区别（或者召回的源是经过了地理位置范围约束后的子集；或者把基于地理位置的召回策略作为最高优先级的策略，其他的召回策略作为它的补充）。

1. 对于**基于热度/流行度的召回**，一般建议按照如下两个因素来组合做召回：

**统计周期**；

**物品/item的顶级类目**

**对于冷启动的用户以及长尾的用户，这个召回策略很有用；即使对于非长尾和冷启动的用户，这个召回策略也是很重要的**。

**之所以这个策略很常用，是基于用户都有一些从众心理的假设**。

该策略经常是首页推荐中的必备召回策略之一。

1. 对于**基于协同过滤的召回**，细分为三种方式：

**基于用户的协同过滤**/UserCF；

**基于物品的协同过滤**/ItemCF；

**基于模型的协同过滤**

**基于协同过滤的方法，本质上是针对用户-物品交互/行为矩阵来建模的**。

只要用户或者Item有变化，甚至action有变化，可能需要重新计算或者重新训练。但是随着这个方法的受欢迎程度越来越大，后来都有了他们的扩展版本即**做增量协同过滤或者近实时协同过滤**（<https://zhuanlan.zhihu.com/p/80069337> ，<https://zhuanlan.zhihu.com/p/80069480> ）

这里并不是说一定要做增量协同过滤或者近实时协同过滤，**可以根据具体业务的用户-物品的交互行为稀疏程度来决定是定期重新计算/重新训练还是使用增量或者近实时协同过滤**

**之所以这个策略很常用，是基于用户的行为共现会影响用户接下来的行为这样的假设**；

该策略经常是首页推荐中的召回策略之一。

对于新闻/资讯/知识类网站一般会使用user-based的CF来召回，由于这些类文章更新太快太多，不适合用item-based的CF（比如头条就用了UserCF）。

对于像电商以及视频网站这类应用一般会使用ItemCF来做召回，因为在这里物品相对来说比较稳定不会变化太快，而且一般相对于用户人数来说也比较少（比如amazon电商以及Netflix视频网站就用了ItemCF）。

1. 对于**基于物品画像的召回**，主要包括下面2种方法：

**基于物品内容的召回**；

**基于物品的整体embedding的召回**

**物品的内容指的是物品的显式属性**：

文本属性可以用文本的embedding来表示，但是属性的embedding区别于物品整体的embedding，基于物品的内容的召回很适合冷启动的物品item。

**物品的整体embedding则是通过某个模型来学习出来的**：

这个模型可以是专门的用户行为序列模型；也可以是更复杂的即包括用户行为序列特征，也包括用户的其他特征和物品特征的模型。

**对于冷启动的物品，可以用某种近似方式来表示它的embedding**。

比如Airbnb使用相近房源的 Embedding 平均值来解决冷启动物品的embedding。Airbnb 的方法是找到 3 个地理位置最接近、房源类别和价格区间相同的已存在的房源，并计算这些房源嵌入的向量平均值来作为新房源的嵌入值。

也就是说物品的画像即包括物品内容这样的显式画像，也包括对整个物品刻画的隐式画像。

**之所以这个策略很常用，是基于物品之间的内涵相似度会影响用户的行为这样的假设；**

该策略经常是首页推荐中的召回策略之一。

1. 基于**用户画像的召回**，这个是最能体现个性化的地方（**个性化推荐中的精髓**）。

一般来说，用户画像包括下面三大类：

**用户的基本人口统计信息**：

比如性别，年龄，常驻的国家/省/市，出生的国家/省/市，学历，职业，收入水平等等-------**注意这些信息的真实性**

**用户的兴趣画像**：

比如用户的兴趣tag，兴趣topic，兴趣关键字，兴趣entity等等；

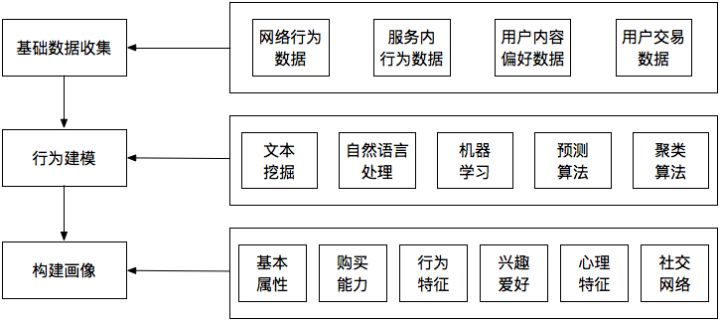
**兴趣画像的一种常见的生成方式**是，首先给物品打上丰富的画像标签，当用户和该物品发生某个行为比如点赞时，就可以把物品的标签比如喜剧类型复制给用户作为用户的一个兴趣tag**。**注意：**如果这里是用户的点“踩”的行为，不方便复制兴趣tag，也不能简单的生成一个“不喜欢戏剧”这样的tag**（因为该用户可能只是不喜欢这个戏剧视频而不是不喜欢所有的戏剧视频），这个时候可以把这个item作为一个很有价值的负样本**，**这个负样本甚至可以做样本加权sample weight。

**用户的行为画像**（这个就是所谓的强特征）：

比如该用户最近N次或者最近某段时间的某个行为的item id列表，刻画用户行为的长期/短期/实时embedding，某个历史统计周期的该用户的某行为的次数，用户的消费水平，某个历史统计周期的该用户的点击率/转化率等等

**具体到做推荐系统的召回的时候，大部分的用户画像是可以用来做召回的**，除了基于历史统计周期的某行为的次数或者比率这样的纯统计特征以及用户的纯行为序列的id列表特征。

而如何构建用户的画像则是重中之重，大致的思路可以参考下图：



**之所以这个策略很常用，是基于每个人的偏好不同并且偏好会反映在行为中的假设。**

该策略经常是首页推荐中的召回策略之一。

而对于“**详情页推荐**”，最通用的召回策略就是**基于item表示向量的相似度的召回**。而item向量如何表示，常见的方法如下（**当然下面的每个方法都可以作为一路召回策略**）：

物品item的显式画像的表示；-------**这个方法对冷启动物品也是有用的**

物品item的整个embedding向量的表示；--------**这个方法最常见**

用户-物品交互矩阵中item对应列向量的表示（假设用户是行，物品是列）；

1. **召回的范式有哪些？针对我们的业务场景的召回范式的选择是什么？**

召回的范式，**需要从三个维度来考虑的**：

**召回策略的数量**；

**单个召回策略下是否有粒度更细分的子策略**？

**召回策略是否分优先级，如果分优先级的话，如何对多个召回策略做融合**？

按照上面的三个维度，召回的范式有如下的可组合的选择：

**单路无子策略召回**;

**单路多个子策略逐层召回**；

**多路无子策略召回**;

**多路多个子策略逐层召回**；

**逐策略无子策略召回**；

**逐策略多个子策略逐层召回**；

**有些业务场景下，某个召回策略是有多个子策略的**。

比如对于基于地理位置的应用，可能有一路召回策略是基于地理位置的召回，子策略可以是附近召回、当前商圈召回、当前城市召回等等，子策略需要根据业务场景来设置优先级。这种情况下，按照业务逻辑来说，先做附近召回，如果附近召回的item数量不够在考虑做当前商圈召回，以此类推。

**对于逐策略来说，也是给每个不同的召回策略分配了不同的优先级**。

召回的时候，优先对最高优先级的策略做召回，如果召回的item数量达到要求就结束召回过程，否则剩下还需要召回的item数量用第二高优先级的策略来做，以此类推。

单路无子策略召回可以用于详情页推荐的场景中。

多路多个子策略逐层召回或多路无子策略召回以及逐策略多个子策略逐层召回或逐策略无子策略召回则常用于首页推荐的场景中（当然，**详情页推荐用多路召回也是可以的**）。

**对于多路召回，在总的召回item数量确定的情况下需要给每一路分配多少item数量**？

1. **不显示区分策略优先级，给每个召回策略分配相同的item数量**（假设总的需要召回的item候选集是1万，可以给每个召回策略分配都是1万，最终通过一个简单的打分粗排模型来选择top 1万的打分结果）。

**这种方法需要用一个模型比如LR逻辑回归模型来学习每路的重要性权重，来对各路召回的结果集打分，最终选择topk的打分结果集送入下游的排序模型**。

对这个模型建模的时候，特征就是一共比如30路的multi-hot特征，对于某个item的多个路的召回，那么对应的这些路就是1，其他路就是0，label就是这个item最后是否被点击/感兴趣。

在推理时候，多路召回的item走一边模型，看最后多路召回的所有不重复item的打分，取分数高的就可以。

这种方法的优点是，每一路都可以给一样的召回条数，最后通过这个模型排序来打分，这个叫为粗排模型也行。缺点就是实现会复杂一些。

注意：在给LR模型积累样本的阶段，如果排序模型已经上线，假设总的需要召回的item候选集是1万，并且一共是10路召回策略，那么这个阶段给每路策略固定1000个item数量（如果考虑有重复的item被召回，那么每路策略可以分配比1000更多一些）；如果排序模型还没有上线（可能在推荐系统起步阶段），那么假设最后的推荐列表是100个，一共10路召回策略，那么可以给每路策略分配10个item（如果考虑有重复的item被召回，那么每路策略可以分配比10更多一些）。

1. **显示设定每个策略的优先级以及每个策略分配的可召回item的数量**。

方法是：**根据最近一段时间统计的某个指标比如点击的召回归因结果来计算出每一路的召回比率**（比如7天内整个大盘点击正样本一共1万，归因到热门召回策略的有5000条，那么热门召回策略的召回比率就是50%），**然后用总的召回结果条数相乘就得到每一路召回条数**（从这里可以看出，**对于每一路的召回策略，设定的召回的item条数是随着时间变化的**）。

注意：召回归因复杂的地方是，从多路策略或者逐策略召回了相同的itemid，这个 itemid最后归属于哪个策略的问题，可选的有两种方法：

一种是多路召回了相同的itemid，那么这个itemid的点击归因到多路上。这种方法的好处是简单直接，直觉上也是合理的。

一种是根据每个策略的历史表现来确定优先级，然后重复出现的itemid会分配给高优先级的那个。

问题来了，**每个策略的历史表现又是如何计算的**？这里一定需要用其他的办法了，比如可以使用上面提到的方法A，也就是不显示区分策略优先级，给每个召回策略分配相同的item数量，然后用一个模型比如LR逻辑回归模型来学习每路的重要性权重。等这个LR模型训练完以后，就得到了每路策略的重要性权重，从而可以确定每路的优先级。

**实际做多路召回的时候，需要考虑在召回的结果不足量（包括因为去重导致的结果不足量）的情况下用高召回率的那路来补足结果**。

因此，实际落地的时候一般每一路都会尽量比要求的结果多召回一些，目的是在可能需要补足的时候不用进行二次召回；

**而对于逐策略召回的话，不需要考虑给每一路分配多少item的困扰**。至于选择多路召回还是逐策略召回，需要具体情况具体分析。

这两种方法各有优缺点，逐策略召回不用考虑每路分配多少条结果的问题，而多路召回相对来说对每路召回更公平一点，也就是可能召回效果更好（**多路召回更多见一些**）。两种范式的具体效果如何，最好还是做线上AB test来对比。

另外，这里会涉到一个相关的问题，**即总的召回结果的item条数如何确定**？

这个没有什么特别的规定，确定的时候最好参考下面的两个因素：

**如果是做线上召回，那么线上召回的latency是否能满足要求**；

**下游的线上排序模型是否可以在latency容许的范围内处理完这么多的item**。

**宗旨是只要上面的两个限定条件能满足，设定的总的召回结果的item条数就应该尽可能多。**

一般来说，**召回的结果去重后需要进行过滤然后在送给排序阶段**，为什么需要过滤这个步骤？

因为存在比如用户把某个item加入黑名单或者显式“踩“过某个物品，为了更好的用户体验，需要维护一个这样的列表，当召回结果去重以后把命中这个list的item也去掉；另外，还可以有一个用户最近一段时间发生过行为的item列表，比如一个小时内点击过的item列表，这样当过滤掉这样最近发生过的item后，客户体验会更好。

**实践中，可能采用bloom filter相关的算法来进行过滤**，这样的性能更高。Bloom filter如果判定一个id在这个集合中不存在那一定是正确的，但是如果判定一个id存在那就不一定是正确了，也就是有误判率，要想很低的误判率就需要更大的bit集合，也就是需要更多的内存。

1. **召回阶段应该使用哪些种类的特征**？

在推荐系统中，我们经常会谈到四大类特征：**用户侧特征，物品侧特征，上下文特征，交叉特征** 。

那是否这四大类特征都适用于用到召回阶段呢？（**这里我们不考虑那些和上下文特征比如地理位置特征强相关的应用，这些应用的召回阶段肯定是需要考虑上下文特征的**）

用户侧特征----------可以用在首页推荐场景的基于用户画像的召回策略中；

物品侧特征----------可以用在首页推荐场景的基于物品画像的召回策略中；也可以用在详情页推荐场景的基于物品item的表示向量相似度的召回策略中；

上下文特征-----------对于不是上下文特征强相关的应用，一般不需要在召回阶段考虑上下文特征。注意：**这里指的是当前上下文特征，而不是用户行为的历史上下文特征**（比如我们在做实时召回模型的离线训练的时候，可以考虑用户历史点击行为序列作为一个特征，点击item时对应的某种历史上下文比如那时使用的APP版本作为另一个序列特征）。

交叉特征-------------包括用户侧特征，物品侧特征，上下文特征这三大类特征之间的特征交叉，以及每类特征内部的特征交叉。一般来说，可能不需要在召回阶段使用交叉特征这么细粒度的特征，这类特征更适合在排序阶段中使用。

其实，这里也能看到之所以**要区分召回阶段和排序阶段的目的**，**除了之前提到的他们的输入的item集合的量级差别很大之外，他们使用到的特征也是不同的**。

1. **每个召回策略返回的topK item集合的依据是什么**？

**排序阶段显然需要个排序的过程，其实对于每个召回策略同样有个排序的过程**，因为每个召回策略对应的源或者说输入的item集合是整个item pool，召回策略选择的是topK的item集合。

**首页推荐场景的不同的召回策略排序的依据也不一样**，具体如下：

**基于热度/流行度的召回策略的排序依据**：

比如最近一段统计周期内物品item加权的平均评分（豆瓣电影网站采用的就是这种）；

比如最近一段统计周期内物品item的点击次数或者转化次数；

**基于物品画像的召回策略的排序依据**：

一种常用的方法：根据用户发生过行为的那些物品画像向量做融合计算出用户的兴趣向量，然后用这个算出的用户的兴趣向量和其他没有发生过行为的物品画像向量计算某种相似度比如cosine相似度。

另一种方式是：把用户喜好作为分类或者回归任务并利用监督学习来建模。构造的每个样本需要包括每个用户的userid以及发生行为的item对应的画像特征（在这个情况下，不需要把itemid放入特征中）。之后预测的时候，直接用模型来给没有发生过行为的item来打分取打分topK的结果。

**基于协同过滤的召回策略的排序依据**：

对于UserCF的方法，首先根据相似度计算出目标用户的邻居集合，然后用邻居用户评分的加权组合来为目标用户作推荐。

对于ItemCF的方法，基于item-item的共现矩阵来计算物品之间的相似度，计算相似度的时候还可以考虑惩罚热门物品以及惩罚活跃用户。

对于基于模型的协同过滤方法，通过User和Item隐向量的点积的值的大小来排序。

**基于用户画像的召回策略的排序依据**：

对于显式画像的召回，一般使用类似热度/流行度召回策略的排序依据；

对于隐式画像即embedding的召回，可以通过计算用户embedding与item embedding的相似度来排序；也可以计算与当前用户embedding相似的邻居用户，然后用类似UserCF中使用邻居用户评分的加权组合的方式来排序，也可以用邻居用户最近点击过的item按照热度来排序。

**详情页推荐的基于item表示向量的相似度的召回策略的排序依据**：

计算详情页物品的表示向量与其他物品的表示向量两者的某种相似度比如cosine相似度来排序。

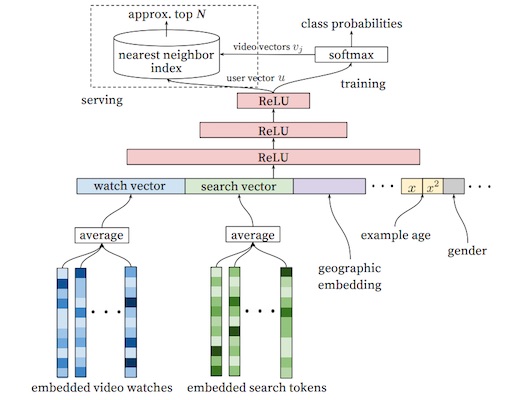
1. **采用离线的召回结果还是做实时召回**？

**实时召回指的是在线上服务时对于每一个访问的用户进行重新计算来召回topK的item集合**。

支持做实时召回的模型一般需要把用户的最近的行为建模进去，这样该模型基于用户最新的行为可以计算得到一个新的用户的embedding向量（显然，实时召回是用在首页推荐场景的）。而那些item向量则是离线用同一个模型计算后并保存在某个高速向量检索库中比如FAISS。

**而离线的召回结果指的是离线把每个用户召回的topK的item集合预计算出来保存到某个NoSQL，线上服务的时候直接从NoSQL中取当前用户的topK的item集合**。

据我所知，**实时召回的流行是从Youtube DNN召回模型（2016年，如下图所示）发布以后开始的**，但是其实类似的工作微软在2013年的DSSM模型中就有。



注意：YoutubeDNN召回模型在离线训练完以后，作者建议把最后一个全连接层的权重矩阵[hiden\_unit, item\_count]做转置后得到的矩阵的每行对应一个video的output embedding，并把这些video embedding存入到一个近邻索引库中比如Faiss，而user embedding则是在线上根据最新的用户行为特征走一遍模型的前向计算取最后一个全连接层的输出的激活值作为他的embedding**。**Example age在训练的时候可能指的是这条样本对应的用户当前这次行为发生的时间离当前训练时间的间隔（paper中作者没有对这个特征讲的很清楚，说加上这个特征模型的效果很好），在线上召回的时候该特征需要设置为0.

**至于采用离线的召回结果还是做实时召回，需要根据业务场景来权衡**。

**实时召回的话，需要更复杂的系统架构支撑**。

比如需要通过一个流式处理引擎来实时获得用户最近的点击序列来更新该用户的点击序列特征并喂给模型。

**离线的召回结果的话，线上系统架构实现简单，缺点是时效性没有实时召回那么好。**

**对于某些业务场景来说 ，可能用户的行为没有那么的频繁，那么用小时级别更新的离线的召回方法就比较适合**。

实际项目中**，经常是有一路是实时召回比如YoutubeDNN召回模型，其他的是多路的离线召回。**

1. **所有的用户/user都走这些召回策略吗**？

**这里分成首页推荐和详情页推荐两个不同的场景来分析**。

1. **对于首页推荐：**

**一般来说对于长尾的用户和冷启动的用户可能走另外的链路效果会更好，而不是所有用户都走同样的召回阶段，排序阶段和重排阶段**。

**对于基于热度/流行度的召回策略，适用于所有的用户**。

**对于基于协同过滤的召回策略，不适合长尾用户和冷启动用户**。

由于针对的是用户-物品的交互矩阵来建模，对每个用户来说至少需要一定数量的行为，而长尾用户和冷启动用户的行为很少或者没有，所以不适合。

**对于基于物品画像的召回策略，分情况讨论：**

**如果用的是基于物品整体embedding的召回，那么对于长尾用户理论上也可以使用这个策略**（因为他至少有过一次对item的行为，但是由于行为还是很少，所以用户embedding的表达可能不准，召回效果可能就不好），冷启动用户不适合；

**如果用的是基于物品内容的召回**，因为计算过程涉及到用户的行为（至少需要一定数量的行为），因此**对于长尾用户和冷启动用户也不适合**。

**对于基于用户画像的召回策略，分情况讨论**：

**用户的行为画像涉及到稠密的用户行为，因此对于长尾用户和冷启动用户不适合**；

**用户的基本人口统计画像是适用于是所有的用户的**，前提是这些人口统计信息是真实正确的；

**用户的兴趣画像一般可以应用到长尾用户**（因为他至少有过一次行为，可以给他打上兴趣画像了）；但是对于冷启动用户，一次行为都没有，兴趣画像可能就不适用了，当然如果该冷启动用户主动给自己打上了兴趣标签的话，兴趣画像的召回也可以用于这样的用户了。

1. **对于详情页推荐：**

**召回策略都是以物品item为中心的，因此所有用户都可以使用同样的召回策略**。

1. **如何评价某个召回策略的好坏**？

召回阶段并不是独立存在的，它往往只是整个推荐系统流程的第一步，后面还有排序阶段和重排阶段；而且召回阶段可能还包括多个召回策略。因此对于召回阶段中的某个召回策略没有什么直接的评价，一般也是线上AB test看效果。**如果使用机器学习模型来建模的话，对召回策略的评价除了前面提到的线上评价，还包括离线评价**。（对于比如做实时召回模型的离线评估，一般常用的离线评估指标是AUC）

1. **需要为了将就复杂耗时的排序模型来削弱召回阶段吗**？

**一定不可以这样做，不能因为下游的排序模型复杂而反过来削弱召回阶段**（比如减少总的召回的item的数量）。

**召回阶段的重要性要大于排序阶段，因此要让排序模型来适配召回阶段，因此不能刻意去追求复杂的大的排序模型**。

另外，**在计算广告中复杂的大的排序模型可能并不适合在推荐系统中的排序模型用**。

比如美团的DPIN深度位置交叉网络模型（用来对位置做debias的模型）是用于广告中的排序模型，**它是针对召回后的候选集集合以及全部曝光位置联合建模**，所以对于推荐系统可能不适合。

因为推荐系统的召回后的结果集一般会比较大比如千级别，而广告召回的结果集一般都很小比如不超过100；而且曝光位置的数量他们可能也有数量级的差别。