# 特征工程

-------------------------------------------------------------------------------

***为什么说， 用物料的后验消费数据做召回存在“幸存者偏差“？能将这些消费数据用于排序吗？***

辨证的看，一个物料的后验指标好，只能说明推荐系统把他推荐给了合适的人，并不意味着把它推给任何人都能取得这个好的效果。这里面存在“幸存者偏差“；另一方面，如何这些后验指标参与精排，”幸存者偏差“影响不会很大，毕竟交给精排模型的物料已经经过了召回、粗排的筛选，或多或少适合当前用户相关的，这些物料的后验指标还是有参考意义的。

***使用物料的后验消费数据做召回， 会放大“马太效应”， 对新物料不友好， 如何缓解？***

后验指标好的物料会被排的靠前，获得更多的曝光与点击，后验指标会更好，形成正向循环。新的物料的后验指标不好甚至没有，排名靠后而较少获得曝光机会，后验指标迟迟得不到改善，形成负向循环。（如何缓解？冷启动问题？）

-------------------------------------------------------------------------------

***解释什么是bias特征？你能举出哪些bias特征的例子？***

我们认为用户点击与否反映了用户正式的兴趣爱好，但严格来讲，以上假设并不成立。实践中，我们无法让所有物料在一个决定公平的环境中供用户挑选，这也就意味着用户的选择并非完全处于他的兴趣爱好，用户的点击未必是他喜欢的，未点击的不代表用户就一定不喜欢。这种不可避免引入的不公平因素叫做bias。例如1. Position bias：“视频5”是用户喜欢的，但是排名靠后，没有点击，将其作为负例就是bias。2. 视频年龄 = 当前时间-上传时间，上传早的后验指标更好，排名靠前，上传晚的没有积累起足够好的后验数据，排名靠后。

***bias特征怎样接入模型？能否和其他正常特征一起喂入DNN底层？为什么？***

解决方式是：1. Above Click, 只有位于被点击物料上方的未点击物料才被纳入负样本。2. 将偏差特征通过一个线性层接入模型，使得模型有足够多的信息来给用户反馈找出合理的解释，绝不能和其他正常特征一起喂入DNN，之后这样才能保证预测时无论伪特征的取值如何，都不会改变排序结果。P 24

-------------------------------------------------------------------------------

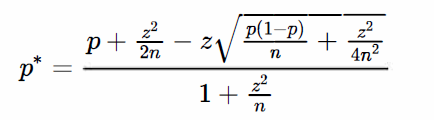
***某男性新用户对“体育”这个分类的喜好程度未知， 如何填充？***

最常规的做法是那所有样本在特征上的mean或者median替代。再精细些，用所有男性用户对“体育“类视频的喜好程度来填充。更合理的做法是训练模型来预测缺失值，利用比较容易得到的人口属性（性别，年龄等）预测新用户对某个内容分类、标签的喜爱程度。

***某新物料的后验指标未知， 如何填充？***

最常规的做法是那所有样本在后验指标上的mean或者median替代。更合理的做法是训练模型来预测缺失值。对于新物料，利用物料的静态画像（分类，标签，品牌，价位）预测它的动态画像（CTR, 平均观看时长，平均销售额）。

***某个物料曝光2次， 被点击1次， 如何计算它的CTR?***

使用“威尔逊区间平滑”“ 

p = 0.5, z = 1.96, n = 2 -🡪 p\* = (0.5+(1.96\*1.96)/2\*2 – 1.96 \* sqrt(0.5\*(1-0.5)+(1.96\*1.96)/(4\*2\*2))) / (1+(1.96\*1.96)/2) = 0.094

-------------------------------------------------------------------------------

***有一个特征“某文章过去1天的点击率是10%", 如何将其构建成一个类别特征， 并喂入推荐模型？***

统计所有的文章过去一天的点击率，排序后，将整个值域的N个分位数（percentile）作为个桶的边界，确保落入各个桶的样本数大致相同。或者单独使用所有的点击率，与目标值拟合一棵决策树，然后，将每个点击率喂入决策树，这个值落入的最终节点的编号就是离散化的结果。

使用 Feature Hashing方法，将特征名字符串转化成Embedding下标，从Embedding矩阵中取得特征的embedding后，通过与其他特征做pooling或者 attention后，进入CTR模型。

# Embedding

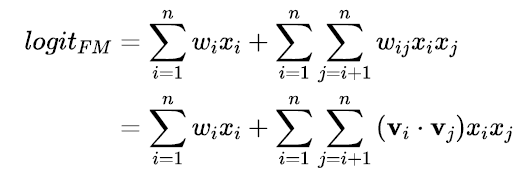
-------------------------------------------------------------------------------

***为什么说Embedding提升了推荐算法的扩展性？***

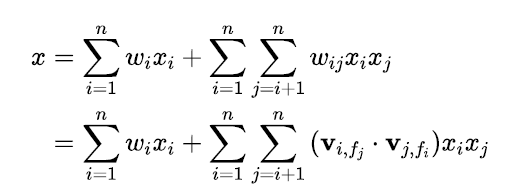
Embedding将推荐算法从“精确查找”变成“模糊查找”，让模型能够“举一反三”。以“科学”与“科技”两个词为例，查找包含“科学”这个词的文章，如果使用不进行同义词扩展，无法召回包含“科技”这个词汇的文章；但使用embedding进行计算，因为“科学”和“科技”这两个词的语义相关度较高，也能召回包含“科技”这个词的文章，这样就扩展了召回的规模。

***FFM针对FM的改进在哪里？***

FM中，每个feature与其他feature交叉式，使用的是同一个embedding，即embedding是共享的。这可能导致相互干扰问题，例如，模型调整Vi 为了把Wij学好，但是可能对另外一对特征组合的稀疏造成负面影响。参数量是n (featrue总数) \* k (Embedding size)



FFM的改进在于，每个特征和不同特征交叉时，根据对方特征所属的Field，使用不同的Embedding。如下公式所示，fj代表特征j所属的Field。Vi,fj 表示第i个feature针对第j个feature所属Field的embedding。说明第i个feature在和属于不同Field的feature交叉式，使用了不同的Embedding。参数量是n (featrue总数) \* f(feature所属filed数量) \* k (Embedding size) 。



***简述阿里Co-Action Network的基本思想？***

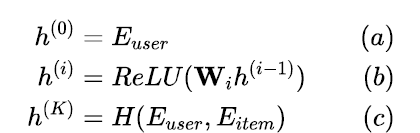
目标是：希望像FFM一样，让每个特征和其他特征交叉时，使用不同的embedding；避免引入FFM那么多的参数。

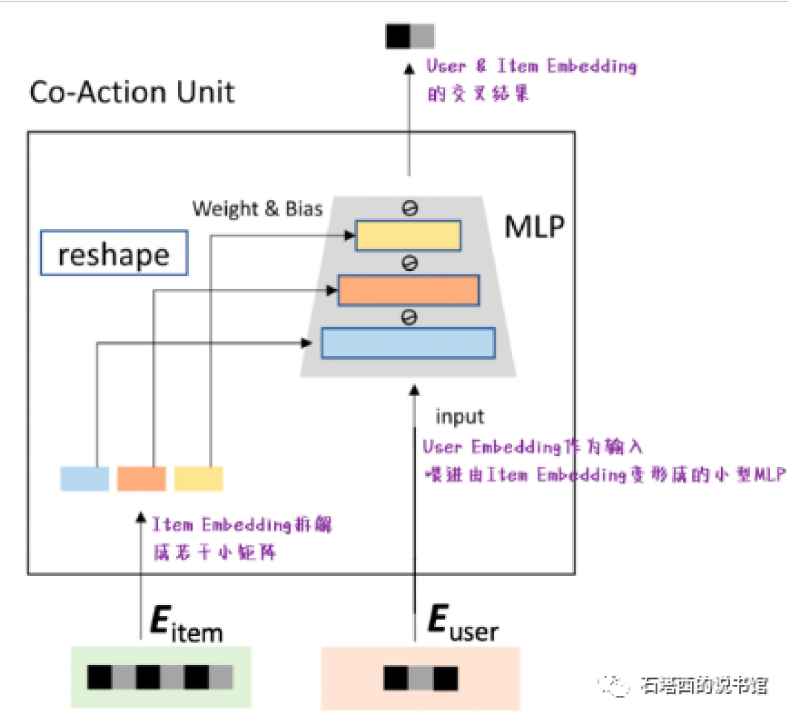
方法：1. 将物料embedding拆成K段，把这K段组织成一个MLP，使用ReLU作为每层的激活函数，将用户特征embedding喂入这个由物料特征embedding变化出来的MLP，MLP的输出就是这两个交叉的结果。

2. 使用ReLU作为激活函数，使得每一层的输出，有一些位置为0，也就是Wi中有一些神经元没有发挥作用；也就是物料特征的embedding和不同的用户特征交叉时，向量中的不同区域发挥作用；也就是同一个物料特征和不同的用户特征交叉时，使用了不同的embedding。

3.没有像FFM那样引入过多的参数，







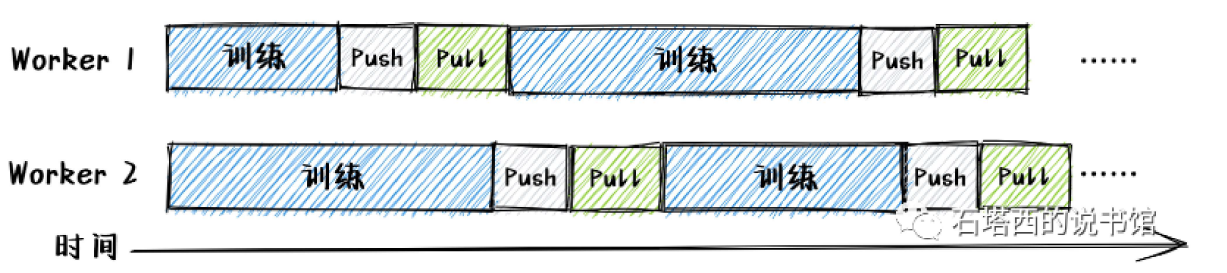
-------------------------------------------------------------------------------

***简述Parameter Server是如何应对推荐系统“高维稀疏”的数据环境的？***

因为特征空间“高维稀疏”的特点，1. Worker节点先汇总当前batch中非零特征，再向Server提出pull参数请求，Server也只回复请求的这些特征（一介特征，Embedding）减少了数据网络传输。2. Server聚合汇总所有Worker发来的梯度，更新相应的参数。

***什么是异步并发(ASP) 中的梯度失效问题？即使如此， 为什么在推荐系统中仍然常用？***

ASP指的是每个Worker推送自己的梯度后，直接开始下一轮Batch训练，不用等待其他Worker。



由于缺乏同步机制，ASP存在“梯度失效”问题（Stale Gradient）,影响收敛速度。例如：

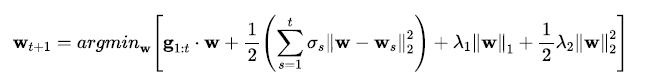
1. Server端有一个参数A\_0, Work\_0和Work\_1同时请求了这个参数，得到后，开始训练。
2. Work\_0训练的快，结束后向Server推送梯度g\_1.
3. Server接受到g\_1后，使用SGD，更新 A\_1 = A\_0 -lr \* g\_1.
4. Work\_1 训练结束，推送g\_2.
5. Server接收到g\_2后，如果使用SGD，这样更新 A\_2 = A\_1llr \* g\_2，就有可能损害收敛。因为g\_2是基于A\_0得到的梯度，但Server只能根据A\_1进行更新，g\_2其实已经失效了。

因为推荐系统的特征高维稀疏，在一轮迭代中，每个Worker分配的训练数据中，非零且重叠的很少。多个Worker同时跟新同一个特征的参数（一阶权重和Embedding）的可能性非常小，在Server端产生冲突的可能性也非常小，所以ASP在推荐系统中十分常用。

# 精排

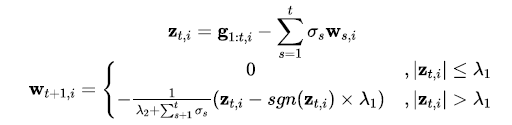
-------------------------------------------------------------------------------

***FTRL是如何保证”在线学习＂的稳定性的？***

******

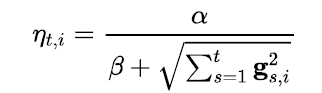
参考上面的代理损失函数，为了减少单个样本的随机扰动，第t步的最优参数，不单单是最小化第t步的损失，而是让之前所有步骤的损失和最小。

***FTRL是如何保证解的稀疏性的？***



当时，第i个特征的新权重Wt+1,i会被直接设置成0，这就是FTRL算法能够产生稀疏解的原因。

***FTRL是如何解决高维稀疏特征受训机会不均匀的问题的？***



FTRL为每一个特征设置独立步长，公式如上：

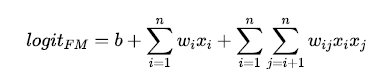
Gs,i 代表由第是个样本算出来的，针对第i个特征的梯度。

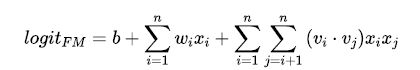
频繁出现的特征，很多样本都贡献了对他的梯度，西格玛累积的比较大，因此步长伊塔会小一些，不会导致已经学的很好的wi发生剧烈变化

较少出现的特征，之前累计的梯度还比较少，步长伊塔会大一些。因为对于超级稀疏的特征，每一个样本都非常珍贵，大补步长有利于对这些样本的应用。

-------------------------------------------------------------------------------

***FM相对LR的优势在哪里？不能只回答自动交叉，是如何解决交叉特征太稀疏、受训机会少的问题的？是如何提高扩展性的？***

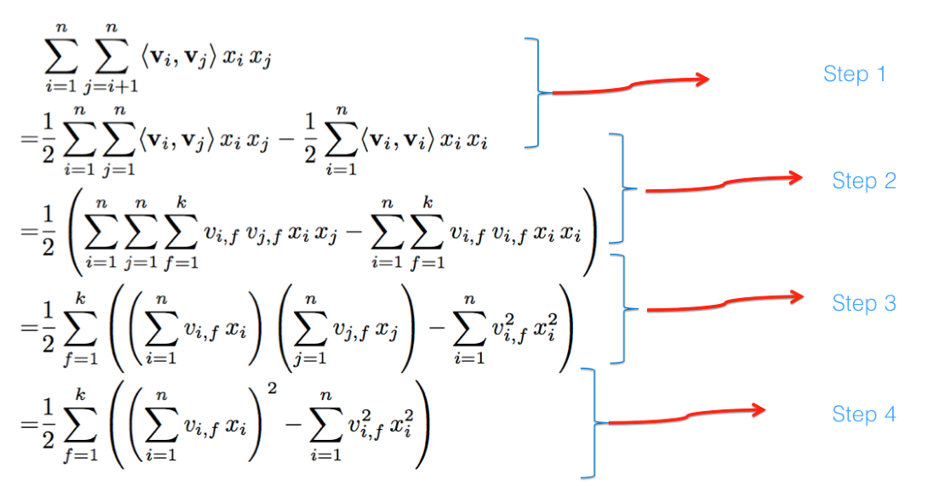
***手动交叉LR***

***FM***

1. 对比LR，FM引入了2阶特征交叉。
2. 学习的参数从n^2变成nk，k为每个特征的embedding长度，减少了参数量。
3. 本来遇到Xi, Xj 都不为零的样本，wij才能得到一次训练机会。但在FM中，只要Xi <> 0的样本都能训练Vi, Xj <> 0 的样本都能训练Vj, 也就间接训练了Wij. 数据利用率更高，训练更加充分。
4. 扩展性：虽然Xi,Xj 这种组合从来没有在训练样本中出现，但Xi, Xj 在训练样本中都单独出现过，因此模型训练好了Vi, Vj. 因此在预测时，FM能够预估出Wij = Vi\*Vj, 从而给小众模式提供一个发挥作用的机会。

***FM对所有特征两两交叉， 岂不是O(N2 )的复杂度？回答是的， 这道题直接fail。回答不是， 要追问FM的实际复杂度是什么？如何实现的？***

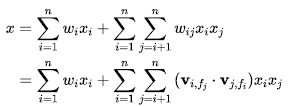
复杂度从O(K\*N^2)，变成O(K\*N). 看下面的推导。

******

***FM的缺点有哪些？提示：其中一个缺点是不灵活， 怎么理解？***

1. 只有2阶交叉特征，没有更高阶的交叉特征。
2. 对所有的2个交叉特征进行了学习，也可能包含无效的信息。
3. 每个特征和其他特征交叉时，使用同一个向量，会造成互相影响。

***FFM相对千FM的改进有哪些？为什么要这么改进？***



FM中，无论特征i 与哪个特征交叉，都使用相同的i来生成交叉特征的系数，这可能储存在相互干扰的问题，比如模型调整Vi以便把Wij学好，但是却可能对另一对他正组合的系数造成负面影响。

FFM中，每个特征在和不同特征交叉时，根据对方特征所属Field要使用不同的embedding。

***FFM相比于FM的缺点在哪里？ （提示：效率）***

FM中，参数数量是N\*K，FFM中，如果这N个feature属于F个field, 参数量是 N\*F\*K，需要更多的训练数据。

-------------------------------------------------------------------------------

***Wide&Deep是如何做到兼顾记忆与“扩展的？***

Wide侧发挥LR”强于记忆“的优势，把训练数据中高频、大众的模式牢牢记住。Wide侧的LR不必像单独LR那样大而全，喂入一些被先验只是认定非常重要的精华特征。

Deep侧对类别特征使用了Embedding以扩展它们的内涵，再加上DNN对特征进行高阶隐士交叉，大大增强了模型的“扩展性“和推荐系统的”多样性“，有利于满足低频、小众、个性化的需求。

***什么样的特征进Deep侧？什么样的特征进Wide侧？***

Wide侧的LR不必像单独LR那样大而全，喂入一些被先验只是认定非常重要的精华特征。主要是人工设计的交叉，共现特征。影响推荐系统的bias特征，例如position bias，只能喂入Wide侧，避免与其他真实特征交叉。

Deep侧，对于类别型特征，将他们的embedding和稠密特征拼接后，喂入模型。

***Wide&Deep论文原文中说， 训练Wide&Deep侧分别使用了两种优化器， 你觉得有哪些道理？***

为了保证Wide侧解的稀疏性，Wide测一般采用FTRL优化器。

Deep测采用DNN常规优化器，比如Adagrad, Adam等。

-------------------------------------------------------------------------------

***DCN解决的是什么问题？***

可以指定任意阶数的显示交叉，补充Deep侧。

***DCN vl和v2的差别在哪里？***

V1

V2认为V1中每一层需要学习的参数只有W和b两个维向量，参数有限。在V2中，用d\*d 的矩阵代替了V1中d维的向量。

V2

D一般很长，每层d\*d的计算压力大，于是将d\*d的矩阵分解成两个 d\*r r\*d 的小矩阵相乘的形式，W=U\*V(t)。将每一层学习的参数量从 d^2减少为2\*d\*r.

V2

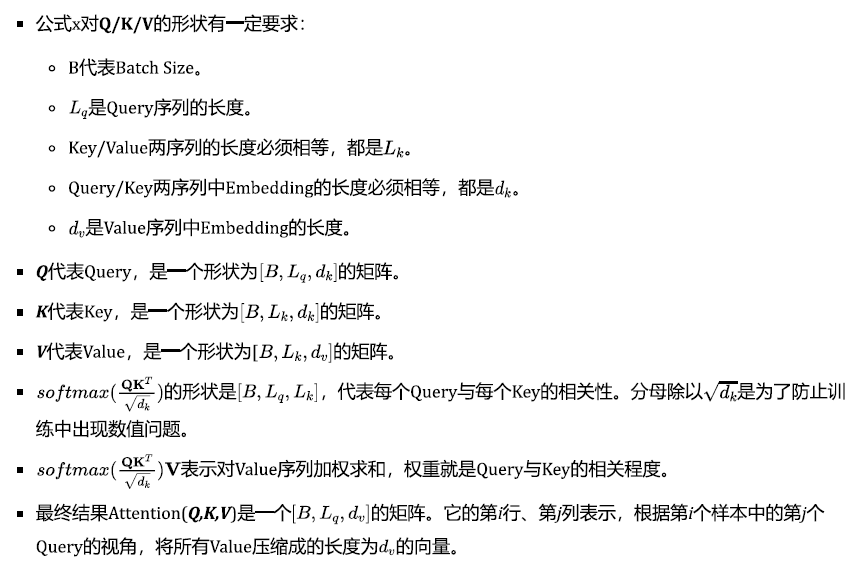
***DCN有哪些缺陷？（提示：输入输出的维度）***

1. 原始输入d很大，每层Cross Layer的参数量不少，所以输入一般都经过挑选的潜在重要特征，不能把所有特征都扔进去。
2. 每层输出都是d维，只做信息交叉，不做信息压缩提炼。

-------------------------------------------------------------------------------

***简述基于Transforme做特征交叉的原理？***

******

******

将所有Field对应的embedding拼接成一个向量，Q=K=V=这个向量。Softmax之后，计算出每个特征(Query)和其他特征(Key)的相关性。再\*V，表示对所有特征（V）序列加权求和，权重就是特征之间的相关程度。

***Transforme做特征交叉的缺点有哪些？ （提示：输入输出维度、时间复杂度）***

1. 为了使用self-attention,要求Field Embedding的长度必须相等。
2. 每层Transformer对信息只交叉不压缩，每层输出形状都是[Batch\_size, M(Field), d(Embedding size)]。做self-attention的时间复杂度是（M\*M\*d）,推荐系统的M和d都很大，开销不小。
3. 实际工作中，我们进选择一部分重要的特征喂入AutoInt做交叉，当作一个特征交叉模块，并不独立预测CTR.

-------------------------------------------------------------------------------

***你在建模行为序列中的每个元素时，一般会包含哪些信息？如何Embedding?***

以“用户最近观看的50个视频“这个序列为例，其中每一个元素的embedding由以下几部分拼接组成：

1. 由每个视频的ID，得到的embedding。
2. 时间差信息，计算观看该视频的时刻距离本次请求时刻的时间差，将这个时间差桶化成一个整数，再embedding。因为历史序列元素对当前候选物料之间的影响肯定随着间隔时间衰减。必须将此信息喂入模型。
3. 上面是最重要的两个，除此之外，还可以加入视频的元信息（制作者，来源，分类，标签等），和动作程度（观看时长，观看完成度）

***每个用户的行为序列长度不同，如何处理? Truncate很简单，关键是如何解决Padding的问题？不解决的话，两个完全不同的序列，因为被填充的大最的0, 而被模型认为相似。***

序列长度超过maxlen的：直接Truncate。

长度不够maxlen的：对于self-attention，对query中缺失部分，用0标记，组成[bs, 1, 1, seqlen] 的向量， 在query\*key 的结果中，mask为0的列，用-1e9替换，然后再对最后一维（列）做softmax时，忽略每行填充0的位置，这样在计算value权重的时候，就会忽略填充位置的value。

***Target Attention的时间复杂度***

Batch大小为B，行为序列长度为L，Embedding长度为D

用候选物料对整个序列做Attention的时间复杂度为 O(B\*L\*D)。

***Self-Attention的时间复杂度***

Batch大小为B，行为序列长度为L，Embedding长度为D

用户行为序列内部做self\_attention的时间复杂度时 O(B\*L^2\*D)。

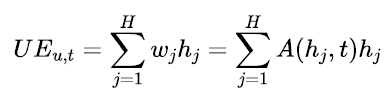
-------------------------------------------------------------------------------

***DIN的建模思路是怎样的？怎么理解“干物干面'?***

从用户行为序列中提取出的兴趣向量应该随当前候选物料的变化而变化，实现“千物千面“的效果。

DIN就是在拿候选物料t当Query对用户的历史行为序列[h1,…hn]做attention, 从而将历史行为序列压缩成一个随候选物料变化的稠密向量，从而实现用户兴趣建模的“千物千面”。

DIN的核心思想是：



UE 是 User Embedding的缩写，下标u和t表示用户兴趣向量取决于用户u与候选物料t两方面的信息。

Hj 表示用户交互过的第j个历史物料的embedding。它可以由该物料的各种属性（商品ID，商店ID，类别ID）的Embedding拼接而成。

Wj = A(hj, t) 是历史物料hj在构成用户兴趣向量中的权重，由当前候选物料t与hj的相似度决定。

A是生成wj的函数，可以简单如点积，也可以复杂为一个小型MLP。

***每到“双十一”之类的促销季， 用户的购买行为与他之前短期行为有较大不同， 应该如何建模？其实就是长序列建模的问题，简单套用SIM不是不可以，但是仍然有效率间题。开放问题，考察候选人的经验，以及思路是否开阔。***

为了性能，参考后面“离线建立用户长期兴趣向量”的回答，但无法“千物千面”。

为了“千物千面”，参考之后SIM的建模思想。

***简述SIM的建模思想？你觉得它的优缺点有哪些？***

使用Attention建模的时间复杂度与用户行为序列长度成线性或者平方关系，L过大时无法满足在线预测和训练更新的实时性要求。但如果建模的序列太短，就无法包含用户的周期性行为，或者包含噪声。

SIM (Search-based interest model)是在线派的代表。在长序列中筛选出与候选物料t相关的一个短序列，在此基础上用DIN。

筛选有两种方法，一种是“硬搜索”，通过两层hash，找出用户长期历史中与其有相同属性的历史物料。一种是“软搜索”，用候选物料的item embedding,在用户长期行为序列中通过近似最邻-ANN, 筛选出候选集。Item embedding可以通过w2v，双塔，等得到。

优点：1. 面对不同的候选物料，筛选出来不同的子集，生成的兴趣向量也不同。

缺点： 实现复杂，在线预测耗时长，训练更新耗时长，只能适用于候选集规模有限的“精排”环节。

***如果想在召回或粗排中建模用户长序列，怎么做？***

离线将用户的长期兴趣挖掘好，缓存起来供在线模型使用，缺点是：得到的长期兴趣向量不会随着候选物料而变化，无法做到“千物千面”。

1. 人工统计长期兴趣，例如每个用户针对某个商品分类或者视频标签在过去一周，一个月等较长时间段内的CTR。
2. 离线预训练一个辅助模型，输入用户的长期行为序列，输出一个embedding代表长期兴趣。
   1. 构建方法：用同一个用户的长期行为序列预测他的短期行为序列。
   2. 使用双塔模型，样本使用三元组：<LS-A, SS-A, SS-B>, LS-A为用户A的长期行为序列，SS-A为用户A的短期行为序列，SS-B为随机采样的用户B的短期行为序列。
   3. 训练过程，LS-A喂入左塔，得到用户A的长期兴趣向量UL-A；SS-A, SS-B喂入右塔，得到用户A和B的短期兴趣向量US-A, US-B。
   4. 建模的目标是：统一用户的长短期兴趣向量应该相近，即cosine(UL-A, UL-B)越大越好，不同用户的长短期兴趣向量应该相距较远，即cosine(UL-A, US-B)越小越好。
   5. 训练好之后，离线将用户们的长期行为喂入左塔，就得到表示个用户的长期兴趣向量。