双塔改建计划

1.1 让user和item更有效地交互

双塔分离：成也萧何，败也萧何

“部署时分离”成了双塔最大的优点，而“训练时分离”成了制约双塔性能的最大因素。

双塔召回模型最直观的缺陷是：

- 特征受限，**无法使用user-item交叉特征。**
- user 和 item 分开建模，最后只通过一次内积来交互，不利于 user-item 交互的学习。最后参与交叉的 user/item embedding，**已经是高度浓缩的了。**一些细节信息已经损失，永远失去了与对侧信息交叉的机会。
- 为了线上快速serving，交叉只能是简单的dot或cosine。一些复杂的、依赖于底层信息的交叉结构，比如target item对user action history的**attention**，也在双塔中找不到位置。

因此双塔第一个优化的中心思想是：**保留更多的信息**在单塔输出向量中，从而有机会和让user和item进行充分交叉

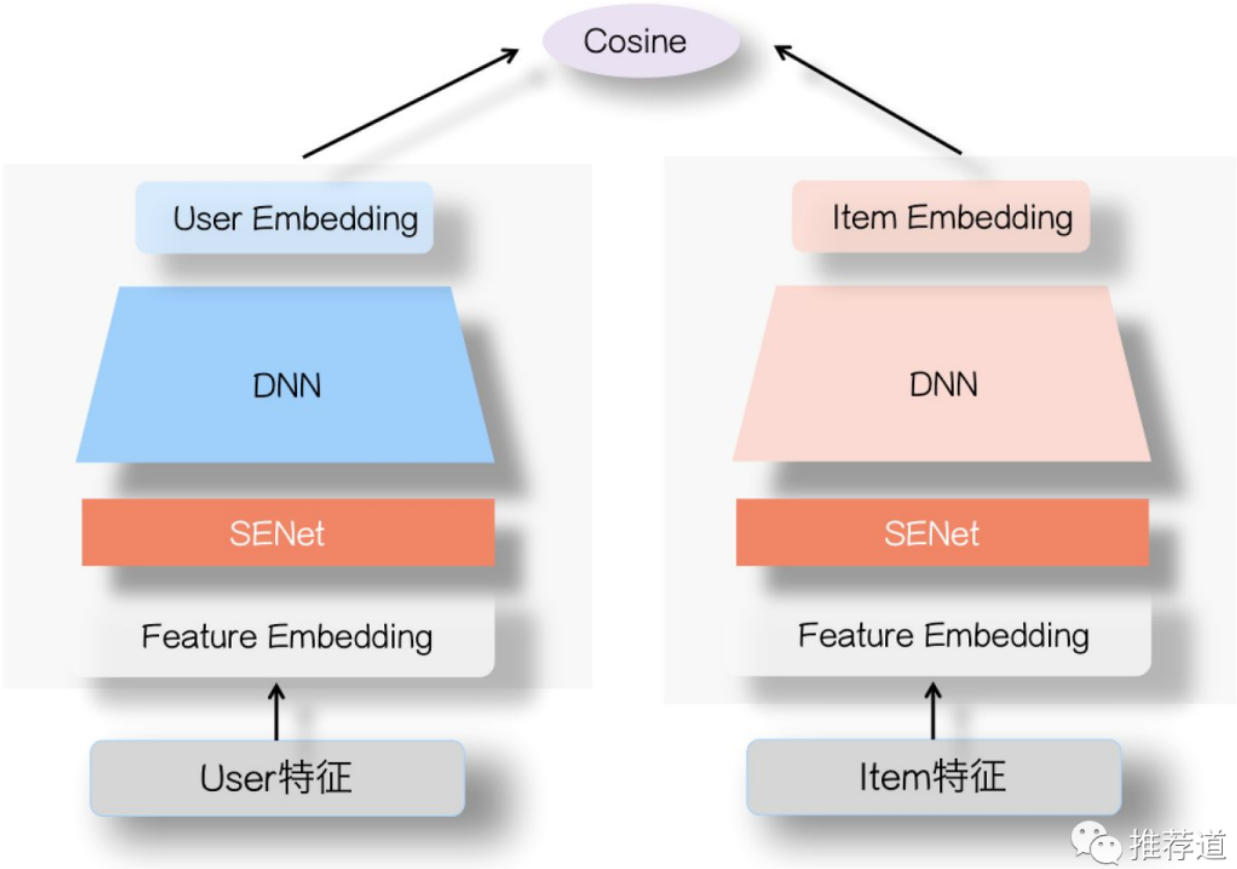
-

综上所述，“双塔分离”的结构，既是保障线上快速serving的优点，也是**不能使用交叉特征与结构、导致两侧信息交叉过晚、制约模型表达能力的最大缺点。**“线上快速serving”正好对召回、粗排这种“大候选集”场景的胃口，而由于后面还有能力强大的精排，所以“模型表达能力弱”的缺点，也能够为召回、粗排所容忍。因此，双塔模型成为召回+粗排的主流模型，几乎是粗排的不二选择。

双塔改建最重要的一条主线就是：如何**保留更多的信息**在tower的final embedding中，从而有机会和对侧塔得到的embedding交叉？围绕着这条主线，勤劳的互联网打工人设计出很多的改进方案。

1.1.2 双塔重地，闲人免进

这种思路以张俊林大佬的SENet为代表。既然把信息“鱼龙混杂”一古脑地喂入塔，其中的噪声造成**污染**，导致很多细粒度的重要信息未能“幸存”到final dot product那一刻。SENet的思路就是，在将信息喂入塔之前，插入SEBlock。SEBlock动态学习各特征的重要性，增强重要信息，**弱化甚至过滤掉原始特征中的噪声，从而减少信息在塔中传播过程中的污染与损耗**，能够让可能多的**重要信息“撑”到final dot product那一刻。**



1.1.2 重要信息，走捷径，一步登顶

信息在塔中向上流动的过程，也是一个信息压缩的过程，不可避免地带来信息损耗。所以，我们很自然地想到，**何不让那些重要信息抄近路，走捷径，把它们直接送到离final dot product更近的地方。**

提到抄近路，大家自然而然地想到ResNet，如下图所示。

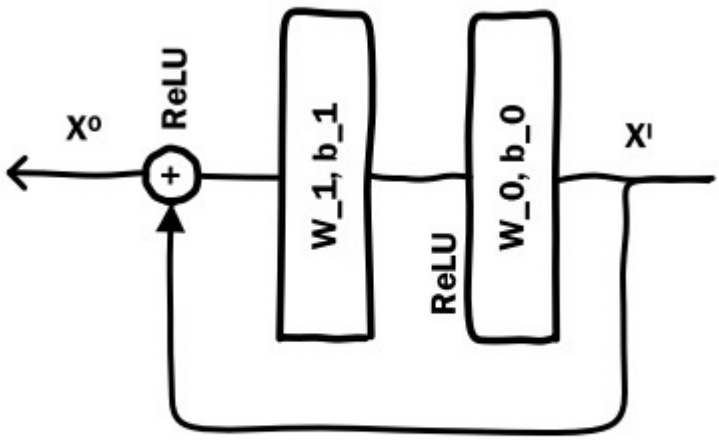


Figure 2: A Residual Unit

- 是喂入塔的原始信息，经过塔中的信息流动，到最后一层时已经损失了很多重要的、细粒度信息。

- 这时，我们将抄近路，送到最后一层与tower的输出融合（图中是element-wise add，但是显然那并不是唯一的融合方式），得到final embedding
- 这时的**既包含了经过tower高度浓缩后的信息，又包含原始输入中的一些细粒度信息。**特别是这些细粒度的重要信息，终于有了和对侧信息交互的机会。

抄近路的思路确定了，那么抄近路的方式，就五花八门，多种多样了。比如除了原始输入能够抄近路，塔中间的一些信息是不是也能抄一把？比如下图模样；另外，既然信息在塔中流动过程中就已经损失了，重要信息没必要等到最后一刻再补充，补充到**中间层**也会大有帮助，就像马拉松选手的中途补水。

但是，**这种抄近路的方式，也有其固有的缺点，就是会导致输出层的肿胀。**比如原来tower final embedding是64维，你现在要将一些重要的、细粒度的信息也抄近路到最后一层。既然称这些信息是细粒度的，自然是未经过压缩提炼的，维度一般都很大，比如1024维。如果你将原来tower embedding与抄近路信息简单拼接，那么final embedding就会膨胀好几倍，会给线上存储、内存都带来巨大的压力。当然你可以将抄近路的信息，经过一层简单的**线性映射**，压缩到一个比较小的维度，但这也会引入额外的映射权重，严重时会导致训练时Out Of Memory。

所以，将所有原始信息无脑地抄近路，显然是行不通的。这就牵扯到另外一个问题：**哪些信息值得抄近路**？要回答这个问题，你当然可以跑一个SE block或其他什么算法，获得各特征的重要性。而从我的个人经验来看，我们要特别注意那些“极其个性化”（e.g., userId, itemId）的特征，和，对划分人群、物群有**显著区分性**的特征（e.g., 用户是新用户还是老用户？用户是否登陆？文章所使用的语言，等）。

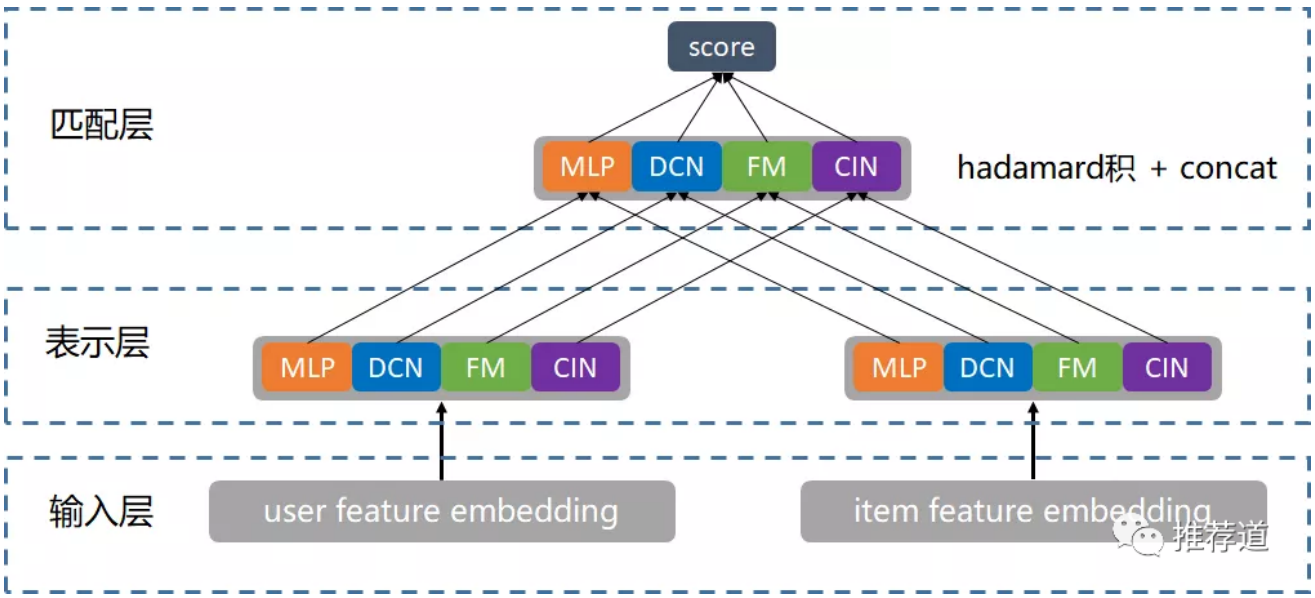
1.2.3 条条大路通塔顶

这一思路有两个出发点：

- 第一个出发点与SENet是相同的：原始双塔，将所有信息一古脑地塞入一个塔，造成向上流动的信息通道拥挤不堪，各路信息相互干扰。
- - SENet的解决方法是“堵”，在喂入塔之前，就将**噪声弱化**甚至屏蔽掉，使塔内的信道变得宽敞，保证重要信息无损通过。
  - 而另外一种思路是“疏”，**大家没必要都挤一个塔向上流动，不同的信息（甚至是相同的信息）可以沿适合自己的塔向上流动浓缩，避免相互干扰。最后由每个小塔的embedding聚合成final embedding**，与对侧的final embedding做dot或cosine。
- 另一个出发点，与“抄近路”的思路是类似的：**我们不再相信（或者说，迷信）DNN的拟合能力。**
- - 传统双塔，只有一种信息上升通道，就是DNN。我估计很多同学有与我类似的经历，就是刚接触DNN的时候，听过这样一句话，“只要DNN足够复杂，能够模拟任意函数”。现在看来，这句话的可信性要大打折扣了，Google DCN的论文里宣称，**"People generally consider DNNs as universal function approximators, that could potentially learn all kinds of feature interactions. However, recent studies found that DNNs are inefficient to even approximately model 2nd or 3rd-order feature crosses."**
  - 既然如此，我们也就没必要将宝都押在DNN这一种通道上。即使是相同的信息，也可以沿多种信息通道向上流动，最终将各通道得到的embedding聚合成final embedding，与对侧交互。

这种思路的典型代表，就是腾讯的并联双塔，“**通过并联多个双塔结构增加双塔模型的宽度，来缓解双塔内积的瓶颈从而提升效果**”

- 信息沿着MLP, DCN, FM, CIN这4种通道向上流动。每种通道各有所长，比如MLP是implicit feature cross，FM和DCN都属于explicit and bounded-degree feature cross，大家相互取长补短。
- 最终各通道的融合，这里是直接**拼接**，只不过各通道的embedding乘上一个可学习的系数，以形成一个logistic regression的效果。这样的话，两边的embedding做点积，就相当于每个小embedding做点积，然后做加权求和嘛。

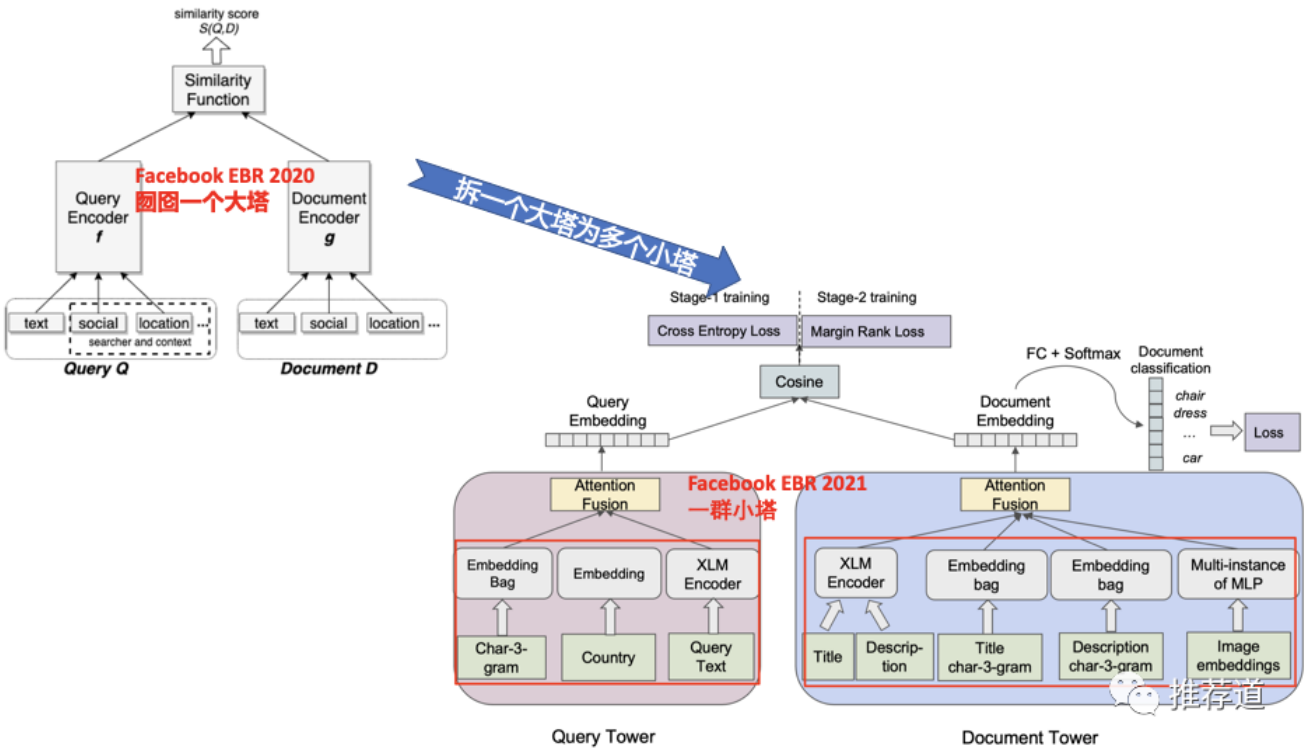


另外，涉及到并联双塔训练细节的是，

- 由于FM和DCN等结构，只能完成信息交叉，而无法信息压缩，所以只能喂入有限的重要特征，否则会引发维度膨胀。
- **虽然不同结构可能共享特征，但是它们却不共享这些特征的底层embedding。**同一个特征，如果要同时喂入MLP和DCN，就必须定义两套embedding，供MLP和DCN分别加以训练。根据我之前的经验，分离embedding空间的确能够换来性能上的提升，但是也带来模型膨胀，给线上serving带来压力。

然而，这篇并联双塔通过"大杂烩"这种"暴力出奇迹"的方式，单纯是增加参数换一些精度，能否打平线上serving**机器资源的增加**也是未知数，原文对比效果的时候甚至都没有打平参数量对比，而原有模型可能增加emd维度本身就能带来auc提升。而且，既然不共享底层embedding，直接训四个召回效果是不是也差不多，而且这个LR就等价于调召回weight嘛。

这种"多塔"思路的另一个代表，来自Facebook的今年最新论文《Que2Search: Fast and Accurate Query and Document Understanding for Search at Facebook》。这篇文章可以算是Facebook 2020年经典论文《Embedding-based Retrieval in Facebook Search》的后继。对比两篇论文的结构图，可以清晰看到"**拆一个大塔为若干小塔**"的思路变化。在Que2Search中，**不同信息通过不同通道向上传递**，比如country这样的categorical特征直接embedding，而文本信息则通过XLM。不同通道得到各自的embedding，再**融合**（fusion）生成final embedding，与对侧塔得到的final embedding计算cosine similarity。



而在融合多塔embedding生成final embedding时，**Que2Search也提出Simple Attention Fusion方案，并通过实验证明，比传统的concatenation+mlp方案有效。**Simple Attention Fusion的方案如下图所示，其中表示第i个通道得到的embedding, 'f'是各通道融合后的final embedding。

$$\varphi = \{\varphi_i\}_{i=1}^{N} \qquad \text{representations of N channels}$$

$$\Phi = \varphi_1 \,\|\, \ldots \,\|\, \varphi_N \qquad \text{concatenation}$$

$$a = \text{Softmax}(\Phi W) \qquad W \in \mathbb{R}^{ND \times N}$$

$$f = \sum_{i=1}^{N} a_i \dot{\varphi}_i \qquad \text{final tower representation}$$

1.2.4 对面的塔儿看过来

之前的几种双塔改建方案，都是针对传统双塔"**交叉太晚**"这一缺点，目标是**让更多有效信息"幸存"到final embedding里，"撑"到final dot product那一时刻**，并通过**净化输入、重要信息走捷径、拓宽信息上升通道**等手段来实现这一思路。以上都是"亡羊补牢"的作法，也就是承认双塔就是交叉太晚，然后想方设法减轻这一缺点带来的信息损失。而美团在2021 KDD上发表的最新论文《A Dual Augmented Two-tower Model for Online Large-scale Recommendation》则选择和"交叉太晚"这一难题正面硬刚。我没有复现并尝试美团的思路，不过，我的确觉得这是一个非常有意思的想法，值得借鉴。

美团的"对偶增强双塔"的出发点是：

- 传统双塔不是交叉太晚吗？那我就发挥深度学习"无中生有"的优点，在user侧"造"出一个embedding模拟item tower的输出，并作为特征接入user tower的最底层；
- 同理，在item侧"造"出一个embedding模拟user tower的输出，并作为特征接入item tower的最底层；
- 这样一来，相当于item tower的输出成为了user tower的输入，user tower的输出成了item tower的输入，两侧信息从一开始就发生了交叉，交叉大大提前了。

具体实现简单描述如下。要了解详情的同学，请移步美团的原文

- user塔的最底层输入: $z_u = [e_{253}||e_{sh}||e_{male}||\ldots||a_u]$,
  - 前边几项比较常规，代表是userId(e.g., 253)、地域(e.g., 上海)、性别(e.g., 男)等特征的embedding的拼接。
  - 最后的a_u就是user侧**增强向量**，通过user id在一个embedding matrix中查询得到，**代表了来自item tower侧与该user进行过正向交互的所有items的信息**
- item塔的最底层输入：
  - 前边几项比较常规，代表是itemId(e.g., 149)、价格(e.g., 10元)、类别(e.g., cate)等特征的embedding的拼接。
  - 最后的就是item侧的**增强向量**，通过itemId在一个embedding matrix中查询得到，**代表了来自user tower侧与该item进行过正向交互的所有users的信息**
- 由原始输入/生成final embedding /的过程就比较传统了，就是分别喂入两侧的MLP
- "对偶增强双塔"的关键是如何学习好两侧的增强向量和，为此作者设计了Adaptive Mimic Mechanism辅助loss来专门训练它们。如下图所示，**公式只在y=1时才发挥作用，目标是让，与所有与之正向交互过的item的final embedding 尽可能接近**，反之对有类似作用。

$$loss_u = \frac{1}{T} \sum_{(u,v,y) \in \mathcal{T}} [y\mathbf{a}_u + (1-y)\mathbf{p}_v - \mathbf{p}_v]^2$$

$$loss_v = \frac{1}{T} \sum_{(u,v,y) \in \mathcal{T}} [y\mathbf{a}_v + (1-y)\mathbf{p}_u - \mathbf{p}_u]^2$$

通过以上方式，“对偶增强双塔”中的每个塔，学习到了对侧信息，并将其作为本侧塔的底层输入，从而**使双塔之间从底层就发生交叉。**

评价：

其实，美团的这种方式，有点蒸馏学习的味道。另外，**要将“与某用户/物料交互过的物料/用户信息作为特征”，恐怕也不必这么麻烦，只要将user action list中的item id先embedding再pooling就可以。**从论文介绍来看增强向量更多地是表征用户的历史点击序列/物品的历史点击用户，那么是否可以将这两个序列特征进行简单池化or聚类就有不错的效果，即结构的收益也许可以通过特征来近似拿到。且论文一共有4个loss，直观感觉训练难度会比较大。但是，真正有意义的是美团这种"让双塔相互深情对视"的创新思路，对我有非常强的借鉴意义，未来或许用得上。

再见，双塔?

如前所述，双塔模型的最大缺点就在于由于双塔分离，造成不能使用任何交叉特征和交叉架构，使得模型表达能力大打折扣。**既然召回、粗排“苦双塔久矣”，何不鼓起勇气，大声和双塔说再见?**

阿里的粗排算法，简单概括之，就是“粗排=特征筛选+精排”。而且遵循两步走策略：**先训练一个模型，通过正则，筛选出“表达能力+计算耗时”性价比高的特征；再用筛选出的特征，训练出一个能够使用交叉特征+交叉结构的模型，用于粗排。**以上只是我对原理的概括，欲了解详情，请移步阿里论文的原文。

- 其次，双塔模型能够快速得到user & item embedding，即便不用于召回或粗排，也能够为精排提供特征。
- 最后，当有充足的线上资源，我们甚至可以改变目前的“召回→粗排→精排”三层架构。我们可以将“粗排”一分为二，用“双塔”作为“召回层的粗排”，用一个“能使用交叉特征+交叉结构的简化版精排”作为“排序层的粗排”，目标是筛选出更优质的物料喂入精排。

总结

从“**净化输入**”、“**重要信息走捷径**”、“**拓宽信息上升通道**”、“**双塔相互模仿**”等方面实现了这个思路，克服双塔的缺点，提升其性能。

2. 让单路召回达到多路的效果

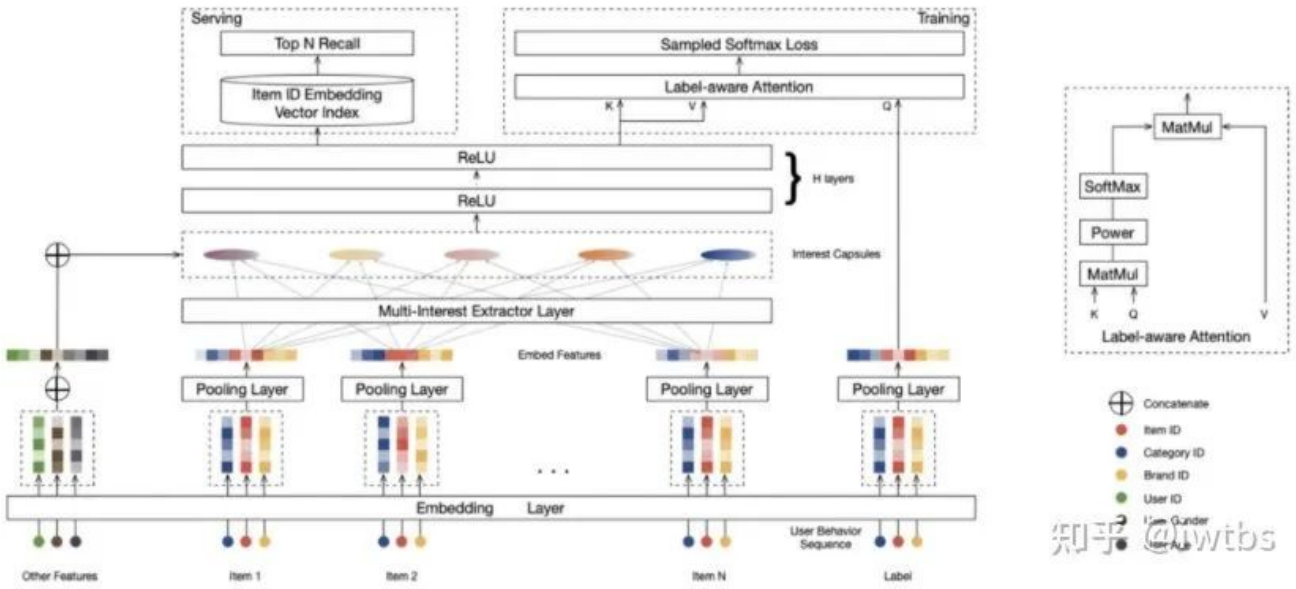
试图使一个模型能够达到多路召回的效果。

2.1 按兴趣域拆分-多兴趣

召回阶段有时候容易碰到头部问题，就比如通过用户兴趣embedding拉回来的物料，可能集中在头部优势领域中，造成弱势兴趣不太能体现出来的问题。而如果把用户兴趣进行拆分，每个兴趣embedding各自拉回部分相关的物料，则可以很大程度缓解召回的头部问题，同时用单路达到多路的效果。

MIND通过引入 capsule network 的思想来解决输出多个向量 embedding 的问题，Multi-Interest 抽取层负责建模用户多个兴趣向量 embedding，然后通过 **Label-aware Attention 结构**，计算target item对多个兴趣向量的注意力，并使用注意力进行加权 -- 这是因为多个兴趣embedding 和target item 的相关性肯定有差异。这样算出来的得分做sampled softmax。

线上 serving时，用户的每个兴趣向量embedding依次通过 KNN 检索得到最相似的 Top-N 候选商品集合。



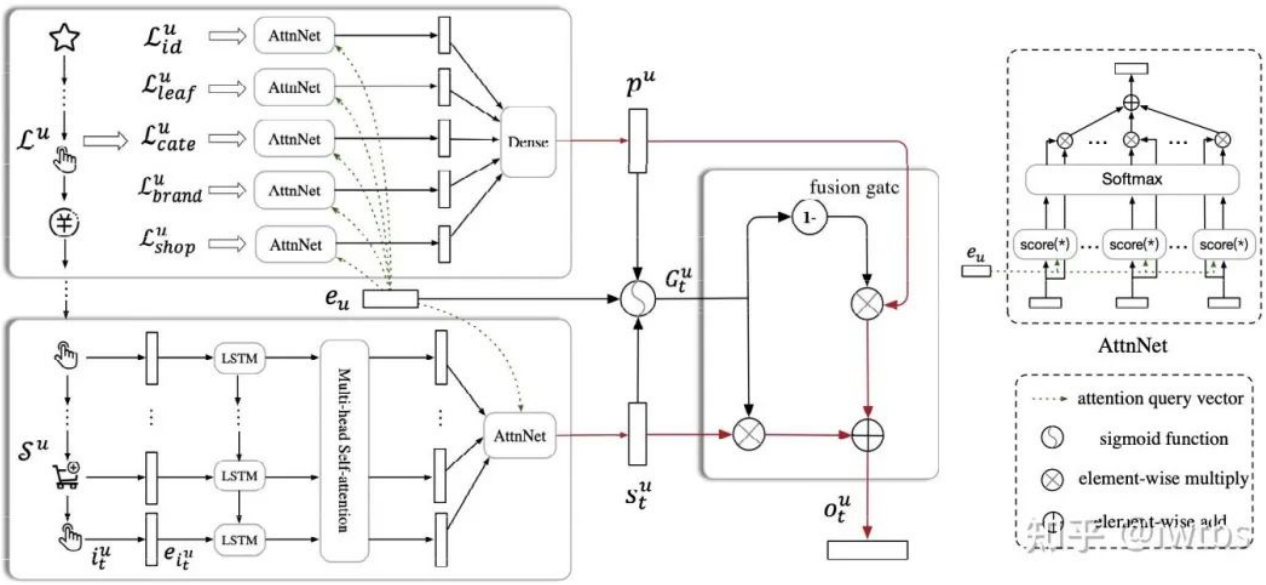
评价：多兴趣确实是一个能拿收益的方向，据说MIND已经成为了手淘召回**占比最高的触发通路。**（召回也是有权重的，不同路的召回数量都是不同的）

2.2 按时间跨度拆分-长短期兴趣

历史行为序列通常可以反映用户短期的兴趣，而长期兴趣则可能随着模型的学习逐渐忘却。在精排阶段我们可以通过超长行为序列建模如SIM建模用户的长期兴趣，而对于时延敏感的召回而言超长行为序列是无法使用的。通过**同时建模用户短、长期兴趣**，用单路召回达到多路的效果。

在《SDM: Sequential Deep Matching Model for Online Large-scale Recommender System》中，

- 作者将**最近一个session定义为短期行为**，利用**LSTM+multi-head self-attention+用用户embedding来做attention**来捕捉短期兴趣序列；
- 发生在**最近一个session前7天内的session定义为长期行为**，通过用户注意力层+DNN得到用户长期兴趣。最终利用fusion gate来控制长短期的影响。（其实也可以对短期兴趣做一个向量、长期兴趣做一个向量，让他们分别去召回？）



评价：实时、短期、长期兴趣建模也是一个被验证能拿收益的方向。

4. 面向后链路的一致性建模

以视频推荐场景为例，一般排序模型会综合多种目标进行综合打分，比如a * ctr+b * staytime + c * 点赞 + d * 转发 + e * 关注...，如果我们只搞ctr、staytime、点赞等召回，就造成了召回和排序目标不一致，显然不利于全链路的优化。

4.1 样本变换

可以基于各种目标进行样本加权，比如同样是点击样本，时长较长、点赞较多的给更大的权重，方法简单直观可解释性强，缺点就是又又又一大堆超参数要调。

4.3 人工构造样本

以百度的《MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search》为例。传统多层的漏斗，首先召回时保证一定的相关性，然后通过rank层预估ctr，ctr较高的内容最终得以展现依然是召回排序目标不一致，导致match层召回的很多广告cpm不高，不会被展现，从而影响商业收入。mobius的思路是把召回层和rank层融合，兼顾相关性和商业指标。具体思路是人工构造出`低相关性且高ctr`的bad case样本，再通过在模型的目标函数中加上bad case率这个目标来进行学习。

我的评价：只要我们理解"模型的目标是拟合数据的分布"，就能明白fake instance为什么能起到作用。本文的模型工程侧也需要做一些配合，同时因为召回引入了bid信息，后面线上也需要关注生态相关的指标。