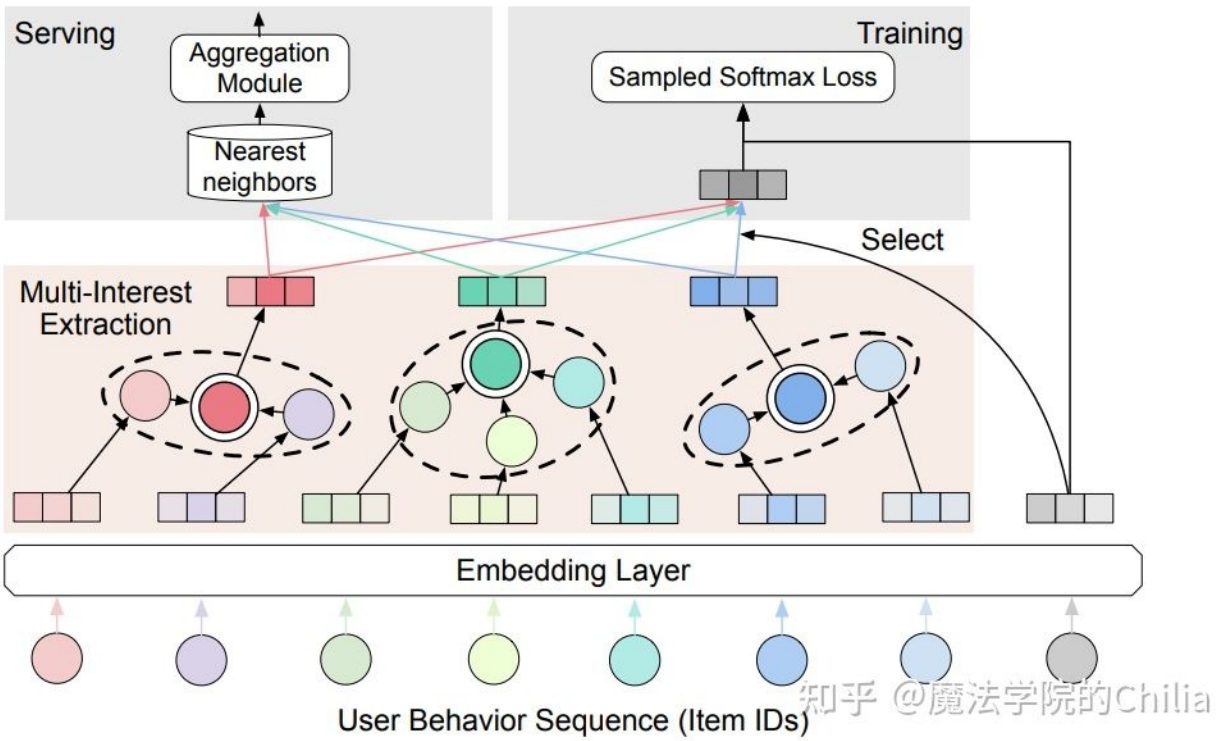
多兴趣 - Comirec

Controllable Multi-Interest Framework for Recommendation

之前我们讲召回的时候讲过用分解user-item交互矩阵的方式得到embedding，但是这种方法数据稀疏的影响，需要的内存也很大；为了解决数据稀疏问题，同时把握住更高阶的user/item关系，可以用图网络进行协同过滤，例如使用GraphSage进行邻居节点的"聚合"从而得到好的user/item embedding，或者通过user-user、item-item相似度构建异构图。但是，使用图网络还是很难融合进来用户的历史行为序列信息。

本文考虑的召回方式可以理解为：从用户的历史行为序列，预测用户下一个交互的item (即**next-item prediction**问题)。



1. 多兴趣表示

本文的第一个重点是，如何从用户的行为序列信息得到好的user embedding。和MIND的思想一样，这里还是要得到用户的**多个**兴趣聚类embedding表示。这是因为用户一般都会有各个方面的兴趣（比如我在b站上看视频的兴趣分类有：历史、游戏、猫猫、美食，每个兴趣点差异很大），用单一向量不能够很好的表示用户的多兴趣，单一向量表征会成为模型的瓶颈。

从用户行为序列得到多兴趣表示的方法有两个：Dynamic Routing和Self-attentive方法。

1.1 Dynamic Routing

和MIND的方法是类似的，都是把底层的向量表示（行为序列）通过加权求和的方式得到上层的向量表示（K个兴趣embedding），其中这些权重是通过迭代的方法进行更新的。

魔法学院的Chilia：阿里推荐召回算法（MIND）：基于动态路由的用户多兴趣网络50 赞同 · 2 评论文章

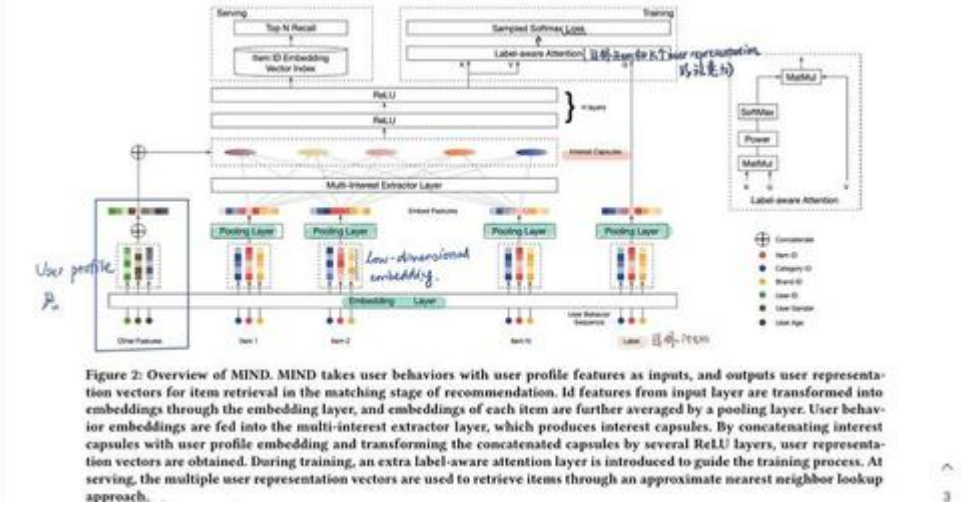


Figure 2: Overview of MIND. MIND takes user behaviors with user profile features as inputs, and outputs user representation vectors for item retrieval in the matching stage of recommendation. Id features from input layer are transformed into embeddings through the embedding layer, and embeddings of each item are further averaged by a pooling layer. User behavior embeddings are fed into the multi-interest extractor layer, which produces interest capsules. By concatenating interest capsules with user profile embedding and transforming the concatenated capsules by several ReLU layers, user representation vectors are obtained. During training, an extra label-aware attention layer is introduced to guide the training process. At serving, the multiple user representation vectors are used to retrieve items through an approximate nearest neighbor lookup approach.

1.2 Self-attentive方法

Self-attentive方法通过注意力机制，计算用户行为序列的加权平均，得到用户的一个兴趣embedding：

$$\mathbf{a} = \text{softmax}(\mathbf{w}_2^\top \tanh(\mathbf{W}_1 \mathbf{H}))^\top,$$

其中n为用户序列长度，d为一个item的embedding size。这里求出来的 a 就是对n个用户序列所加的**权重**。那么，用户的一个兴趣embedding就是以a为权重，对用户序列做的加权求和。同时，为了加入位置信息，可以把position encoding做成d维向量，和item embedding H 相加。

如果想要得到用户的多个兴趣聚类embedding，那么就使用K个转换向量 w_2 ,得到K个概率分布，以此作为权重加权得到K个用户兴趣embedding：

$$\mathbf{A} = \text{softmax}(\mathbf{W}_2^\top \tanh(\mathbf{W}_1 \mathbf{H}))^\top.$$

这样，我们就得到了K个用户兴趣聚类embedding。

2. 模型训练

现在，我们得到了user的K个兴趣embedding，那么怎么把这K个embedding综合成一个呢？在MIND里采用的是比较复杂的Label-aware attention，即用target item作为Query，user的K个兴趣embedding作为Key和Value，做self-attention。但是在这篇文章里就比较简单了，直接取和target item 的 embedding点积最大的那个兴趣embedding即可，就用它来作为user塔的输出，得到了user embedding。

$$\mathbf{v}_u = \mathbf{V}_u[:, \text{argmax}(\mathbf{V}_u^\top \mathbf{e}_i)],$$

对于训练目标，我们还是先进行负采样，然后使用Youtube DNN一文中所述的sampled softmax方法。

训练完成之后，上线模型。当用户发出一个query之后，我们就用模型先计算出用户的K个兴趣embedding，然后每个兴趣embedding召回topN个最相近的item。这K*N个item就将进入后面的aggregation module，通过综合考虑准确率和多样性，得到最终召回的top N个item。

3. Aggregation Module

Aggregation Module用来综合从不同兴趣点召回的物品，实现推荐**准确率**和**多样性**的平衡。

推荐的【多样性】问题：多样性和准确率是trade-off的关系。如果我们只把那些最确定的item推荐给用户，准确率是很高，但是内容同质化严重，用户会觉得十分无聊（信息茧房）。这就是很多人抱怨的："我昨天看了几个美食视频，结果今天就**只**给我推美食"。

推荐系统的多样性分为两类，一是"aggregated diversity", 指的是推荐系统推荐长尾物品的能力，是站在整个推荐系统的角度来讲的；二是"individual diversity", 指的是给一个用户推荐的商品的不相似性(dissimilarity). 这是针对给一个用户的推荐结果来讲的。在这篇文章中，我们来优化的是individual diversity.

上面说到，我们已经有了K*N个item，那么如何得到最后需要召回的TopK个item呢？如果我们只考虑准确率的话，当然是user embedding和item embedding点积得分越大越好，得分是MaxSim，计算如下：

$$f(u,i) = \max_{1 \le k \le K} (\mathbf{e}_i^\top \mathbf{v}_u^{(k)}),$$

这就是item embedding和K个兴趣embedding的最大点积。按照这个得分去给K*N个item排序，然后取topN即可。

但是，我们毕竟还想要兼顾准确率和多样性的。所以，我们要优化的目标（最大化）是下面这个式子：

$$Q(u,\mathcal{S}) = \sum_{i \in \mathcal{S}} f(u,i) + \lambda \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} g(i,j). \quad (12)$$

点积得分 召回物品. controllable factor 两两不相似度

其中，S为我们要找的那TopN个召回物品，第一项就是maxsim点积得分，我们想让user/item相似度越大越好；第二项是召回物品的两两**不相似度**(dissimilarity),我们想要让召回的物品之间差距越大越好。 λ 是controllable factor，控制着准确率和多样性的trade-off，这就是题目中"controllable"的由来。

不相似度的计算方法：

$$g(i,j) = \delta(\text{CATE}(i) \neq \text{CATE}(j)).$$

我们使用**贪婪**的方法来最大化Q(u,S):

Algorithm 2: Greedy Inference

Input: Candidate item set $\mathcal{M}$, number of output items N
Output: Output item set $\mathcal{S}$

$\mathcal{S} = \emptyset$
for $iter = 1, \cdots, N$ **do**
  $j = \text{argmax}_{i \in \mathcal{M} \setminus \mathcal{S}} \left(f(u,i) + \lambda \sum_{k \in \mathcal{S}} g(i,k) \right)$
  $\mathcal{S} = \mathcal{S} \cup \{j\}$
return $\mathcal{S}$

每次贪婪的加入使得Q(u,S)增大最多的那个item

在做评估的时候，也使用"多样性(diversity)"作为一个评估标准：

$$\text{Diversity@N} = \frac{\sum_{j=1}^{N} \sum_{k=j+1}^{N} \delta(\text{CATE}(\hat{i}_{u,j}) \neq \text{CATE}(\hat{i}_{u,k}))}{N \times (N-1)/2}, \tag{17}$$

即归一化的两两不相似度。