Controllable Multi-Interest Framework for Recommendation：可调控的多兴趣推荐框架

目前，神经网络模型已经被广泛地应用到各种推荐系统中，而这种基于神经网络的推荐算法通常只会从用户的行为序列中学习到一个用户表征向量，这个统一的表征向量往往无法反映用户在一段时期内的多种不同的兴趣，因此作者团队提出了一种可调控的多兴趣推荐框架来解决这种情形：多兴趣抽取模块会从用户的点击序列中捕获到用户的多种不同的兴趣，然后用于召回各种兴趣对应的商品，聚合模块会将这些不同兴趣召回的商品整合起来作为推荐的候选商品，供下游任务来使用。

本文为 KDD 2020 ADS Track所接收。

1 论文背景

首先介绍电商推荐系统的一些研究背景：近年来各种电商平台发展得非常的迅速，各大电商平台，比如淘宝、京东其实都有一套各自的个性化推荐系统，大家平时都能在对应的手机APP的首页上看到一些个性化的商品推荐，那么这些业界的推荐系统究竟是怎么做的？怎么来处理电商平台中海量的商品？

1.1 推荐系统的两个阶段

业界的推荐系统主要包含两个阶段，分别是matching和ranking阶段，我们一般称之为召回和排序两个阶段。

第一个是召回阶段，它会从海量的商品集合中召回一些用户可能感兴趣的候选的商品集合，可能会有数百个，然后输入到下游的排序阶段。排序阶段会对这些候选商品进行排序，然后根据一些更加细粒度的用户跟商品的特征给出一个最后的顺序。当然这只是一个大致的流程，实际业界的推荐系统往往会更加复杂，但是学术界往往只关心这样两个阶段，由于这两个阶段的功能不同，两个阶段使用的模型也会有一定的差异。

简单介绍这两个阶段所使用的模型的区别：对于召回阶段，召回模型会接收用户的行为得到相应的用户的表征，然后用户的表征会跟商品的表征进行点积的计算来得到相似度，继而进行模型的训练。值得一提的是这里只能用一些简单的相似度计算方式，比如内积，因为在后续的召回的时候，需要进行快速的 Top k近邻召回。典型的召回模型大概有YouTube DNN、GRU4REC两个模型。

对于排序阶段，我们通常会将这一部分任务叫做CTR预估任务，排序的模型可以同时接收用户的行为和商品的表征，然后进行一些特征的交叉来得到更好的精度。它的训练方式是通过预测用户是否点击来进行训练，典型的模型有Wide&Deep，DeepFM等。

1.2 K近邻检索

召回阶段会通过用户的表征来从海量的商品集合中检索k近邻的商品集合，所以这里需要一个快速的检索算法。

在论文中使用了Faiss库，这是一个 Facebook AI开发的一个基于稠密向量的高效检索算法，它的具体使用方式包含两部分，首先通过商品表征来构建一些索引，接着就可以直接通过用户的表征向量直接来快速的检索出相似的商品。值得一提的是该库支持GPU计算，检索效率比较高。论文中提出的方法，主要是针对召回阶段所设计。

2 论文解决的问题

很多推荐算法他们只考虑了用户单方面的一个兴趣，但是其实每个用户他其实在一段时间内他都会有很多不同的兴趣。

比如这样一个例子，一个电商用户他可能有很多兴趣，包括珠宝类，手机包类以及化妆品三个类别，然后我们的模块会从它的点击序列中学到这样三种不同的兴趣，然后每种不同的兴趣，又可以从海量的商品集合中召回出近相似的一些商品，那么我们之后还会用一个聚合的模块，来将这些不同兴趣召回的商品对它们进行整合。然后接着我们通过一种调控的方式，使得输出的商品它的精度和多样性都能够得到比较好的保证。

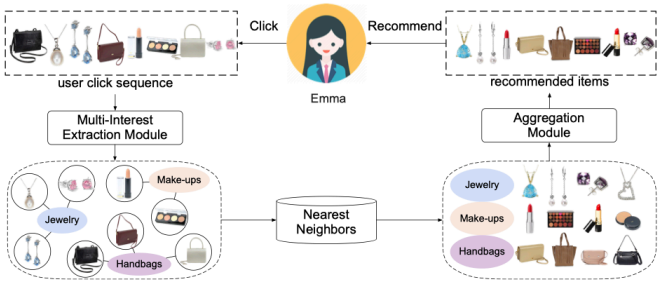

Figure 1: A motivating example of our proposed framework. An e-commerce platform user, Emma, has multiple interests including jewelry, handbags, and make-ups. Our multi-interest extraction module can capture these three interests from her click sequence. Each interest retrieves items from the large-scale item pool based on the interest embedding independently. An aggregation module combines items from different interests and outputs the overall top-N recommended items for Emma.

那么序列化推荐的问题定义如下：假如我们有这样一个用户的集合以及商品的集合，那么对于每个用户我们会有它历史的一些行为，它可能是一些点击的商品的序列，那么他们是通过用户与他们进行交互的时间来排序的，给定这样用户的一些历史的交互数据，序列化推荐就是想预测下一个用户可能会感兴趣，或者可能会对可能会点击的一些商品，这是我们所提出的模型的整体的框架。

3 论文使用的方法

3.1 ComiRec模型

作者团队提出的模型叫 ComiRec，输入侧是用户的一些行为的序列，他们可能是一些商品的编号，这些商品的编号会通过表征层得到一些商品的表征向量。然后多兴趣抽取模块会将这些表征向量聚合起来，聚合出多种不同的兴趣，这多种不同的兴趣可以用来训练，然后进行之后的推荐模型的预测。

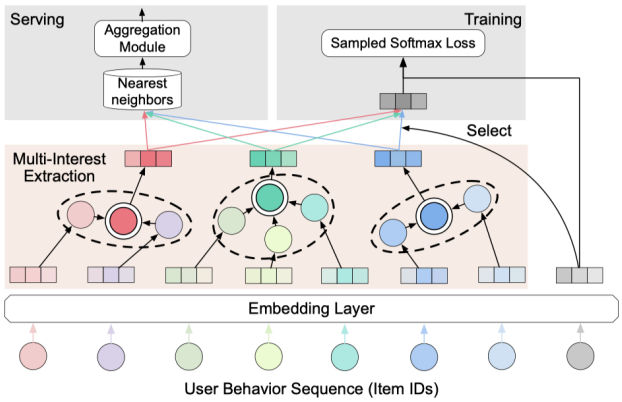

Figure 2: An overview of our model for the sequential recommendation. The input of our model is a user behavior sequence, which contains a list of item IDs. The item IDs are fed into the embedding layer and transformed into the item embeddings. Interest embeddings are generated through the multi-interest extraction module and can be then used for model training and serving. For model training, the nearest interest embedding to the target embedding will be chosen to compute the sampled softmax loss. For serving, each interest embedding will independently retrieve top-N nearest items, which are then fed into the aggregation module. The aggregation module generates the overall top-N items by a controllable procedure that balances the recommendation accuracy and diversity.

在Training中，对于一个给定的目标的商品，会从用户的不同兴趣中挑选一个与其最接近的兴趣，然后对他们进行计算得出loss然后来训练。在Serving中，会对每个不同的兴趣表征都进行k近邻的检索，然后对于这些检索出来的商品再通过一个聚合模块来统一的得到Top-k的推荐结果。在聚合模块中，会通过一种可调控的方式来保证结果的精度跟多样性。

3.2 多兴趣抽取模块

在论文中作者主要探索了两种方式来建立这样一个多兴趣抽取模块，一种是Dynamic Routing，它是由一篇名为"Dynamic Routing Between Capsules"的paper提出的，是一个迭代式的算法，把商品的表征看作是看作是一种primary capsules。对于用户的兴趣，我们可以把它看作是一种兴趣的胶囊，可以使用这种叠加式的方法，来得到用户兴趣的表征。

Algorithm 1: Dynamic Routing

Input: primary capsules $\mathbf{e}_i$, iteration times r, number of interest capsules K

Output: interest capsules $\{\mathbf{v}_j, j = 1, ..., K\}$

for each primary capsule i and interest capsule j: initialize $b_{ij} = 0$.

for $iter = 1, \cdots, r$ **do**

  for each primary capsule i: $\mathbf{c}_i = \text{softmax}(\mathbf{b}_i)$.

  for each interest capsule j: $\mathbf{s}_j = \sum_i c_{ij} \mathbf{W}_{ij} \mathbf{e}_i$.

  for each interest capsule j: $\mathbf{v}_j = \text{squash}(\mathbf{s}_j)$.

  for each primary capsule i and interest capsule j: $b_{ij} = b_{ij} + \mathbf{v}_j^\top \mathbf{W}_{ij} \mathbf{e}_i$.

return $\{\mathbf{v}_j, j = 1, ..., K\}$

还有一种方式是这种基于自注意力Self-Attentive的方式，对于用户的行为的这样一个表征的矩阵，我们可以通过一种自注意的方式得到这样一个注意力矩阵，然后以此来得到用户最后的表征。

Given the embeddings of user behaviors $\mathbf{H} \in \mathbb{R}^{d \times n}$, we use the self-attention to obtain attention matrix $\mathbf{A} \in \mathbb{R}^{n \times K}$

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

Here, n is the length and K is the interest number

The final matrix of user interests are

$$\mathbf{V}_u = \mathbf{H}\mathbf{A}$$

计算完用户的表征之后，我们会通过argmax来选取用户表征中与我们训练样本商品最接近的一个表征，然后构造一个似然值，再通过负对数似然的这样一个loss来训练我们的模型。值得一提的是如果直接去计算loss，因为需要所有的商品所以其实非常耗时，这里通过采样的方式来加速模型的训练。

- Given a training sample (u, i) with user embedding V_u and item embedding $\boldsymbol{e}_i$, we compute the likelihood as

$$P_\theta(i|u) = \frac{\exp(\boldsymbol{v}_u^T \boldsymbol{e}_i)}{\sum_{j \in I} \exp(\boldsymbol{v}_u^T \boldsymbol{e}_j)}$$

where $\boldsymbol{v}_u = V_u[:, argmax(V_u^T \boldsymbol{e}_i)]$

- The objective function is

$$loss = \sum_{(u,i)} -\log P_\theta(i|u)$$

3.3 聚合模块

在模型训练之后，我们可以给每个用户得到多种不同的兴趣表征。然后基于这种。基于不同兴趣表征召回出来的商品，我们需要用这样一种聚合的模块来得到统一的推荐的结果。一种简单并直接的方式，就是说将这些不同兴趣召回的商品按这样一个f函数来进行排序，这是一种比较直接的方式。

- Define the score between a user and an item:

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

- We use $Q(u, S)$ to balance the accuracy and diversity by a factor $\lambda \geq 0$:

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

Here, $g(i, j)$ is a diversity function such as

$$g(i, j) = \delta(CATE(i) \neq CATE(j))$$

那么我们这里考虑的是如何把推荐结果的多样性也考虑进去，那么我们就引入了下面这样一个式子，我们在后面增加了这一项，这一项g(i,j)就代表一种多样性的函数，它表示了i,j这两个商品它们的类别或者其他的属性是否是一致的，如果不一致的话就是1，一致的话就是0。

然后我们在在这个Q函数的后面这一项来调控推荐的多样性，然后这边还引入了这样一个因子，lamda它可以调控两者的比例。最精确的一种结果，就是我们取lamda=0的时候，就相当于我们不考虑后面这一项。那么跟最直接的那种方式就是一致的。当lamda趋近于正无穷的时候，那么其实我们就可以忽略前面这一项，那么我们输出的结果应该是最具有多样性的这样一个推荐结果。

这里我们提出了一种贪心策略，来近似的最优化的这样一个价值函数。贪心的策略就是每一步我们都取当前最大的这样一个值，然后将它输出出去。

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 $\mathcal{S}$

4 论文的结果

实验的一些设置：将所有的用户分为三个不同的集合，以8:1:1的比例划分成训练验证和测试集。对于训练集合，我们每个训练样本会使用用户前k个行为去预测第k+1个行为。在评估过程中，我们通过用户前80%的行为去计算得到用户的兴趣表征，然后去预测剩下20%的行为。

作者使用了这样三种不同的指标来衡量就是模型的性能，第一个是Recall，第二个是Hit Rate，然后是NDCG，其实都是一些常用的推荐或者说信息检索的一些指标。作者选取的一些对比的方法包括 Most Popular，YouTube DNN，GPU4REC以及MIND这4个不同的方法。

作者在这样一个序列化推荐的任务中，设计了一些实验来验证模型的性能。作者使用了两个公开的数据集，一个是亚马逊的，一个是淘宝数据集，约有几十万个用户。根据对比表格显示，在这两个数据集上，作者的模型在多个指标上都要显著的优于其他的方法。

Table 3: Model performance on public datasets. Bolded numbers are the best performance of each column. All the numbers in the table are percentage numbers with '%' omitted.

	Amazon Books						Taobao					
	Metrics@20			Metrics@50			Metrics@20			Metrics@50		
	Recall	NDCG	Hit Rate	Recall	NDCG	Hit Rate	Recall	NDCG	Hit Rate	Recall	NDCG	Hit Rate
MostPopular	1.368	2.259	3.020	2.400	3.936	5.226	0.395	2.065	5.424	0.735	3.603	9.309
YouTube DNN	4.567	7.670	10.285	7.312	12.075	15.894	4.205	14.511	28.785	6.172	20.248	39.108
GRU4Rec	4.057	6.803	8.945	6.501	10.369	13.666	5.884	22.095	35.745	8.494	29.396	46.068
MIND	4.862	7.933	10.618	7.638	12.230	16.145	6.281	20.394	38.119	8.155	25.069	45.846
ComiRec-SA	**5.489**	8.991	11.402	**8.467**	**13.563**	17.202	**6.900**	**24.682**	41.549	9.462	31.278	51.064
ComiRec-DR	5.311	**9.185**	**12.005**	8.106	13.520	**17.583**	6.890	24.007	**41.746**	**9.818**	**31.365**	**52.418**

对于可调控方面，作者定义Diversity是在推荐的集合中，两两商品之间它们的类别的差异。我们可以看到当lamda增大的时候，diversity多样性会显著的提升，而推荐的精度recall只会有一些许的下降，那么当我们选择一个和比较合适的lamda的时候，我们的这样的一个聚合模块能够在准确率和多样性这两者之间取得一定的平衡。

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

Table 5: Model performance of Amazon dataset for the controllable study. All the numbers are percentage numbers with '%' omitted.

	ComiRec-SA (K=4)		ComiRec-DR (K=4)	
Metric@50	Recall	Diversity	Recall	Diversity
$\lambda = 0.00$	**8.467**	23.237	**8.106**	19.036
$\lambda = 0.05$	8.347	38.808	7.931	42.915
$\lambda = 0.10$	8.229	46.731	7.850	46.258
$\lambda = 0.15$	8.142	51.135	7.820	46.912
$\lambda = 0.20$	8.086	53.671	7.783	47.581
$\lambda = 0.25$	8.034	**55.100**	7.764	**48.375**

这是一个电商用户的例子，我们通过训练好的模型，从它的点击序列中挖掘出了用户4种不同的兴趣，左侧我们展示了用户的点击序列中与这4种兴趣相对应的商品。我们可以看到这4种不同的兴趣分别是甜食、礼物和手机壳以及小配件。 右侧是我们通过捕获到的用户兴趣，然后从商品池中召回出来的一些商品可以看到召回出来的商品跟前面的兴趣其实还是比较一致的。

Figure 3: A case study of an e-commerce user. We generate four interest embeddings from the click sequence of a user by our model. We find that the four interests of the user are about sweets, gift boxes, phone cases, and accessories. We report those items in the click sequence that correspond to the four interests. The right part shows the items retrieved from the industrial item pool by interest embeddings.

作者团队在2020年2月8日手机淘宝App采集的行业数据集上进行了进一步的实验，拆分并使用用户的点击序列来训练模型。该行业数据集包含2200万条优质商品、1.45亿用户、40亿条交互行为。在本文框架和最先进的顺序推荐方法 MIND 之间进行了离线对比实验，显示该框架已使得阿里巴巴集团推荐系统得到显着改进。实验结果表明，与 MIND 相比，ComiRec-SA 和 ComiRec-DR 分别将 Recall@50 提高了 1.39% 和 8.65%。

5 未来工作和我的看法

本文是由清华大学KEG实验室和阿里巴巴达摩院的多位学者共同完成的，他们在文章的Indutrial Results部分提到，该框架已成功部署在阿里巴巴分布式云平台上，十亿级工业数据集的结果进一步证实了该框架在实践中的有效性和高效性。

由于深度学习的快速发展，推荐系统开始了一个新阶段。传统的推荐方式无法满足行业需求。对于未来而言，可以计划利用记忆网络来捕捉用户不断变化的兴趣，并引入认知理论来进行更好的用户建模。

推荐系统依然会随着时间的推移和技术的发展有着无限的优化空间，希望我们可以为之做出贡献。