京东语义搜索召回

在实际的搜索应用中，仅仅考虑相关性往往是不够的。尤其在候选量很大、计算资源有限的情况下，我们更希望优先召回高成交率的商品。想要达到这种目标，需要在**语义相关**（Semantic）目标基础上增加**个性化**（Personalization）特征。今天要讲的论文顾名思义，即要设计同时面向个性化和语义的向量检索系统，能够提高**长尾**query的CVR。方案比较常规，基本可以代表目前电商在个性化搜索召回部分的常规做法。论文链接：https://arxiv.org/pdf/2006.02282.pdf

1. 简介

1.1 搜索系统工作流程

- Query Processing: 进行query重写，包括词干化(tokenization)、拼写**纠错**(spelling correction)、query**扩展**（query expansion）. 例如搜索"cellphone for grandpa"，该模块将此query表示为term-based representation：[TERM cellphone] AND [TERM grandpa]. 这样是为了下一步要用倒排索引检索的方便。
- Candidate Retrieval：多路召回，亿级->千级
- Ranking：一般是cascading ranking(粗排->精排)，模型由简单到复杂。

1.2 电商搜索的两个挑战：

- 怎么召回那些和query虽然没有exact match、但是有语义相关性的商品。（不可否认，exact match是搜索召回的重要一路，但是只用倒排索引查exact match还远远不够，还需要增加新的召回路，比如语义向量召回。这里不得不吐槽一下实验室接的美国某二线大厂的项目，他们到现在竟然还只用倒排索引在召回，真的是让人怀疑人生~）一个方法是用query expansion，扩展出一些近义词，但是并不能够保证所有近义词都能够涵盖。
- 增加个性化。每个人搜索出来的结果都不一样，正所谓“千人千面”。

本文提出DPSR(Deep Personalized and Semantic Retrieval), 实现+1.29%的CVR提升，其中**长尾query**提升+10.03%。

2. 相关工作

文章提到了一些经典的信息检索工作，包括：

- 基于矩阵分解的LSI
- BM25
- DSSM, DRMM, Duet。（使用传统lexical matching信号，例如query term importance、exact matching信号）

在实际应用中，离线算好item embedding，训练好query embedding tower，然后用faiss来完成亿级向量相似度匹配，达到千级的QPS(queries per second).

3. 模型

3.1 特征

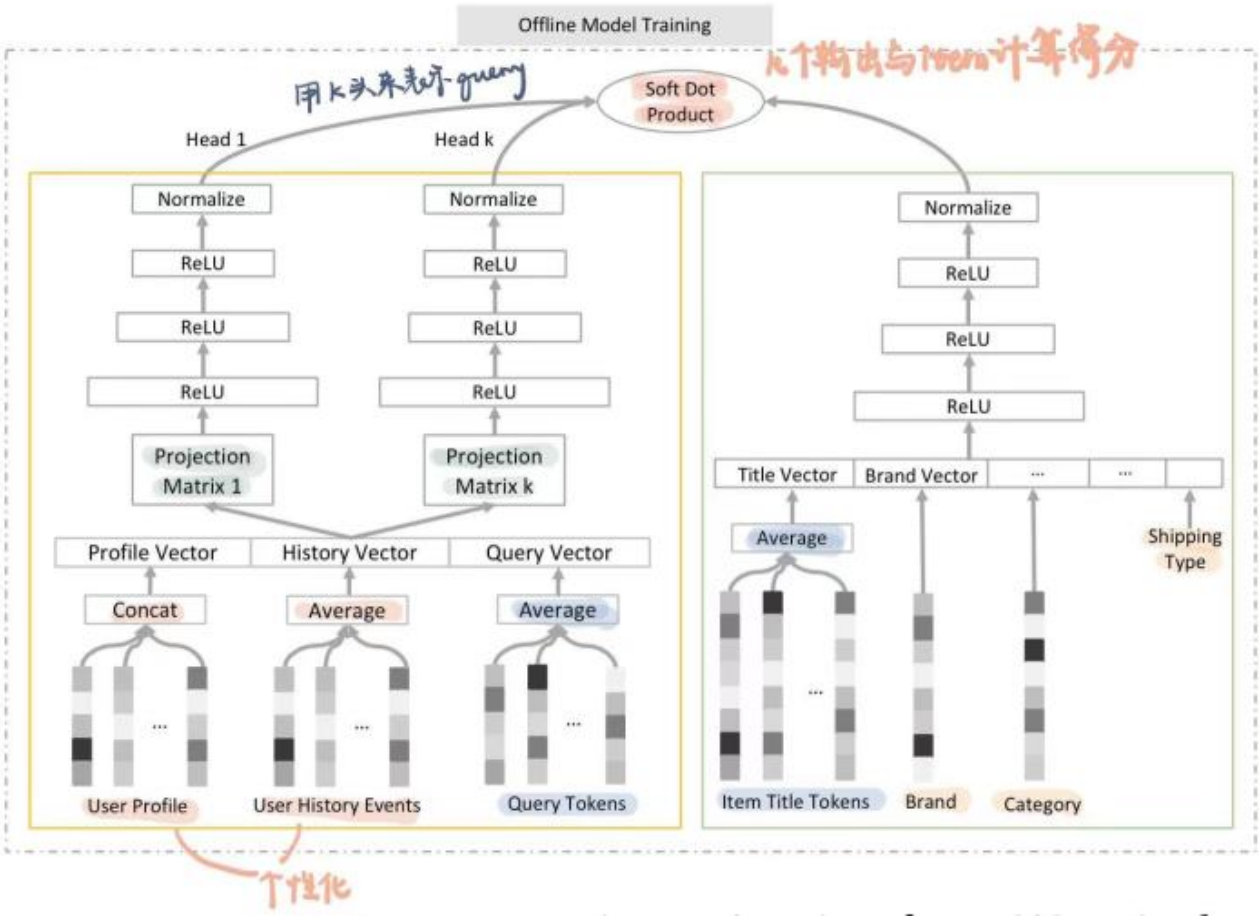
对于query token和item title token做avg-pooling，这是为了节约召回阶段的训练资源。注意这里的"token"应该是指不同粒度的，比如subword-level(BPE)，word-level，n-gram. 这样是为了能够自动完成纠错、把握不同粒度的信息。

本文方案还增加了多种个性化特征以提升模型对个性化信息的关注，用户侧包括：

- User Profile：用户画像特征，如用户性别、年龄、消费能力、区域等，用于刻画用户的静态基础特征。由于特征不多，所以concat到一起就好。
- User History Events：用户历史行为特征，如历史点击商品、搜索query、类目品牌偏好、点击率、成交率等。为了节省计算资源，直接avg-pooling。

商品侧特征包括：商品品类、品牌、邮寄类型，又如商品、店铺历史表现等。通过增加个性化特征，模型能够捕获用户偏好和商品除文本语义之外的属性特征。

文中提到，我们当然可以用RNN、transformer等模型来做序列建模，但是这里只用了MLP,这是因为MLP消耗计算资源少，而且比更复杂的模型差不了哪去。尤其在召回阶段，速度是第一位的！

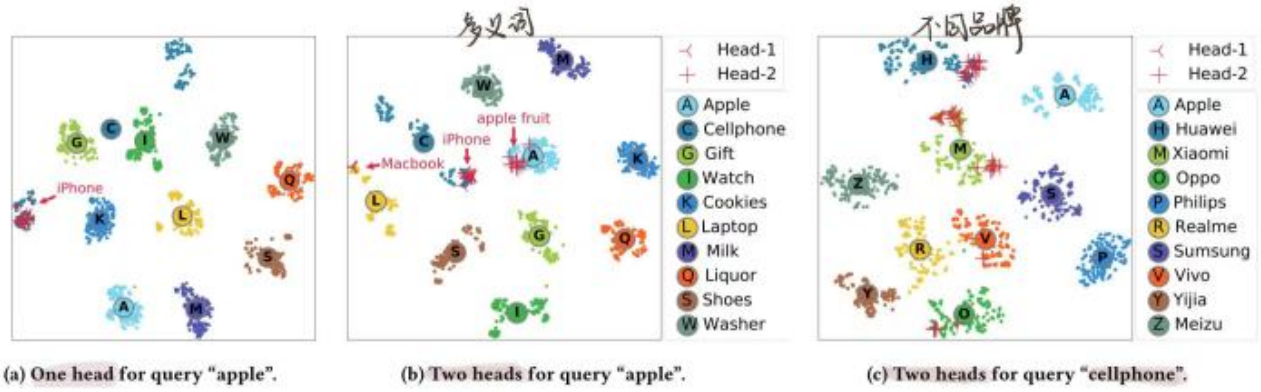


3.2 双塔

模型采用经典的双塔架构，包括user/query tower和item tower两个模块。其中item tower比较轻量级，就是一个多层MLP。

user/query tower就有点不一样了，它使用了**k个【multi-head】来提取更加多样的特征**（就是图中k个不同的projection matrix, 有点模型ensemble的感觉。其实和transformer中的multi-head十分类似，用不同的W_Q, W_K, W_V矩阵生成在不同子空间的特征向量）。这个想法来自于transformer的多头注意力，就像是CNN中不同的卷积核，提取着不同的特征。这样，我们就从提取了query的k个特征，**能够更加全面的表示用户和query。**

多头可以把握住不同的用户intention，例如"apple":水果/macbook/iphone? "cellphone":华为/小米?



(a) One head for query "apple". (b) Two heads for query "apple". (c) Two heads for query "cellphone".

Figure 6: t-SNE visualizations of retrieval results for polysemous queries.

3.3 分数计算

item tower和user/query multi-head使用attention的方式进行匹配**分数融合**：

$$G(Q(q), S(s)) = \sum_{i=1}^{m} w_i e_i^\top g. \tag{1}$$

其中权重的计算：

$$w_i = \frac{\exp(e_i^\top g/\beta)}{\sum_{j=1}^{m} \exp(e_j^\top g/\beta)},$$

3.4 损失函数

click log中只包含click的物品，也就是只有正例。所以负例要用**负采样**来构建。具体怎么负采样会在3.5节中介绍。假设我们对每个query q_i 都找到了一组负样本 N_i,那么对每个query就有一组 $< q_i, s_i^+, N_i >$训练集，既有正例又有负例。那么，**pairwise**损失函数为：

$$\mathcal{L}(\mathcal{D}) = \sum_{(q_i, s_i^+, N_i) \in \mathcal{D}} \sum_{s_j^- \in N_i} \max\left(0, \delta - f(q_i, s_i^+) + f(q_i, s_j^-)\right).$$

注意本文是使用的正负样本对triplet loss，而不是向Youtube双塔一样使用sampled softmax。

3.5 负采样

使用用户点击数据为正样本（10亿级别）。负样本同样没有使用曝光未点击的商品，因为未被点击的商品不一定不相关。负例分为两部分：random negatives、in-batch negatives，二者合并作为负样本参与训练

$$N_i = N_{rand} \cup N_{batch}$$

。

random negative就是从item数据库中随机找的，为了节省计算成本，**一个batch的正样本都共享一组负例** N_{rand}。

in-batch negative是指每个batch中的每个item都做过一次正例，BATCHSIZE-1次负例。一个样本做in-batch负例的概率和它被点击的频率是正相关的（也就是它出现在batch中的概率），所以这种方法会**把热门的商品多当作负例，给热门商品以惩罚**（在Youtube 双塔一文中，就用频率估计的方法解决了in-batch negative对热门商品以不必要的惩罚这一问题。）而random negative中**每个样本被采样的概率都是一样的。**

文章指出随着random negative比例的增加，**更容易召回popular商品**（因为没有给popular商品更多惩罚），更容易点击/成交，相应地**相关性也会一定程度下降。**

4. 总结

这篇文章除了在query tower增加多头之外，在模型上并没有太大创新，但是把语义搜索召回的整个流程说的很清楚，就是业界做搜索召回的一个真实写照。