淘宝搜索中的语义向量检索技术

Embedding-based Product Retrieval in Taobao Search

整体淘宝搜索系统包括四阶段：match-prerank-rank-rerank（召回，粗排，精排，重排），本文重点在于召回。

挑战：和web-search不同，电商平台的文本通常**较短**，没有语法结构；同时要考虑海量的用户历史行为来做个性化。基于lexical matching的倒排索引搜索引擎性能好、可控性强，尽管存在一些语义鸿沟问题，但仍被广泛的应用在现有的搜索引擎架构中。但是，这种搜索引擎无法有效区分**相同query下**， 不同用户的**兴趣差异**，即无法捕捉用户**个性化**的特征。因此，如何**高效**地检索出语义上最相关、且最能够满足用户个性化需求的商品，权衡【query语义】和【用户个性化历史行为】之间的关系，是电商平台主要的挑战。

很多做电商语义搜索的文章只强调在指标上提升很多，却没有说明**向量召回会降低相关性**（因为泛化能力太强，导致记忆能力弱，召回出很多不相关的item），导致用户抱怨的BAD CASE。

作者也部署了基于语义向量的检索系统在淘宝搜索中，观察了很长一段时间，有几个发现：

- 短期效果很好；长期来看，基于embedding的方法由于不是词匹配，即：缺乏【**完整匹配**(exact match)】query所有terms的能力，很容易造成相关性BAD CASE。
- **为了能够保证相关性**，作者采用了一个相关性控制模块，来对检索到的商品做过滤。控制模块对EBR检索后的结果，做进一步的完整匹配过滤，只保留那些能够**完整匹配**结构化字段的商品，给到后续的排序阶段。作者统计了下，这几乎会过滤掉30%的商品，即：30%的商品相关性较低，被过滤的商品既耗费计算资源，又不能够参与到后续精排，导致本来可以进入后续排序的相关性商品无法进入排序，整体指标下降。
- 因此，本文的主要目标是期望基于向量的模型能够检索到更多相关的商品，**有更多相关的商品**能够参与到后续排序阶段中，从而在保证相关性的前提下，提高整个系统的线上指标。

这篇文章的核心贡献总结如下：

- **模型**：提出了一种**多粒度深度语义商品检索模型**(Multi-Grained Deep Semantic Product Retrieval (MGDSPR) Model)，能够动态地捕捉用户搜索语义和个性化交互历史行为的关系，兼顾**语义**和**个性化**。
- **训练和推理的一致性**：为了保证训练和推理的一致性，使得模型具备全局比较能力，除了使用随机负采样外，作者采用了softmax交叉熵损失函数，而不是hinge pairwise损失函数，因为后者只具备局部比较能力。

> During inference, the model needs to select the top-K products closest to the current query from all candidates, requiring the ability for **global** comparison. However, pairwise hinge loss can only do **local** comparison

- **相关性保证**：1.**在softmax基础上引入温度参数**，对用户隐式反馈(点击数据)进行相关性噪声的平滑。2.**混合正样本和随机负样本**来产生"**相关性增强**"的困难负样本。进一步，作者采用了**相关性控制模块**来保证EBR系统的相关性。
- **实验和分析**：在真实的工业级数据上，阐明了MGDSPR的有效性。进一步分析了MGDSPR对**搜索系统每个阶段**的影响。

2. Solution

问题形式化：$\mathcal{U}=\{u_1,\ldots,u_u,\ldots,u_n\}$表示$N$个用户集合；$\mathcal{Q}=\{q_1,\ldots,q_u,\ldots,q_N\}$表示用户相应的queries，$\mathcal{I}=\{i_1,\ldots,i_i,\ldots,i_M\}$表示$M$个物品的集合。同时，作者将用户$u$的历史行为根据离当前的时间间隔划分为3个子集合，

- **实时行为序列**(当前时间戳前的若干行为)：$\mathcal{R}^u=\{i_1^u,\ldots,i_t^u,\ldots,i_T^u\}$；
- **短期行为序列**(不包括在$\mathcal{R}^u$中的10天内的行为)：$\mathcal{S}^u=\{i_1^u,\ldots,i_t^u,\ldots,i_T^u\}$
- **长期行为序列**(不包括在$\mathcal{R}^u$和$\mathcal{S}^u$中的1个月内的行为序列)：$\mathcal{L}^u=\{i_1^u,\ldots,i_t^u,\ldots,i_T^u\}$。

任务：给定用户u的历史行为序列$(\mathcal{R}^u,\mathcal{S}^u,\mathcal{L}^u)$，他在时间$t$发起了一次搜索请求$q_u$，我们期望返回物品的集合$i\in\mathcal{I}$来满足该用户的搜索需求。具体而言，目标是基于用户$(\text{query},\text{behaviors})$和物品**items**之间的得分$z$，从$\mathcal{I}$中预测出Top-$K$候选物品。即：

$$z=\mathcal{F}(\phi(q_u,\mathcal{R}^u,\mathcal{S}^u,\mathcal{L}^u),\psi(i))$$

其中，$\mathcal{F}(\cdot)$是打分函数，$\phi(\cdot)$是query/behaviors的编码器，$\psi(i)$是item编码器。作者也是采用了双塔的召回模型，$\mathcal{F}$用内积函数来表示。

先介绍下**整体的网络框架结构**：

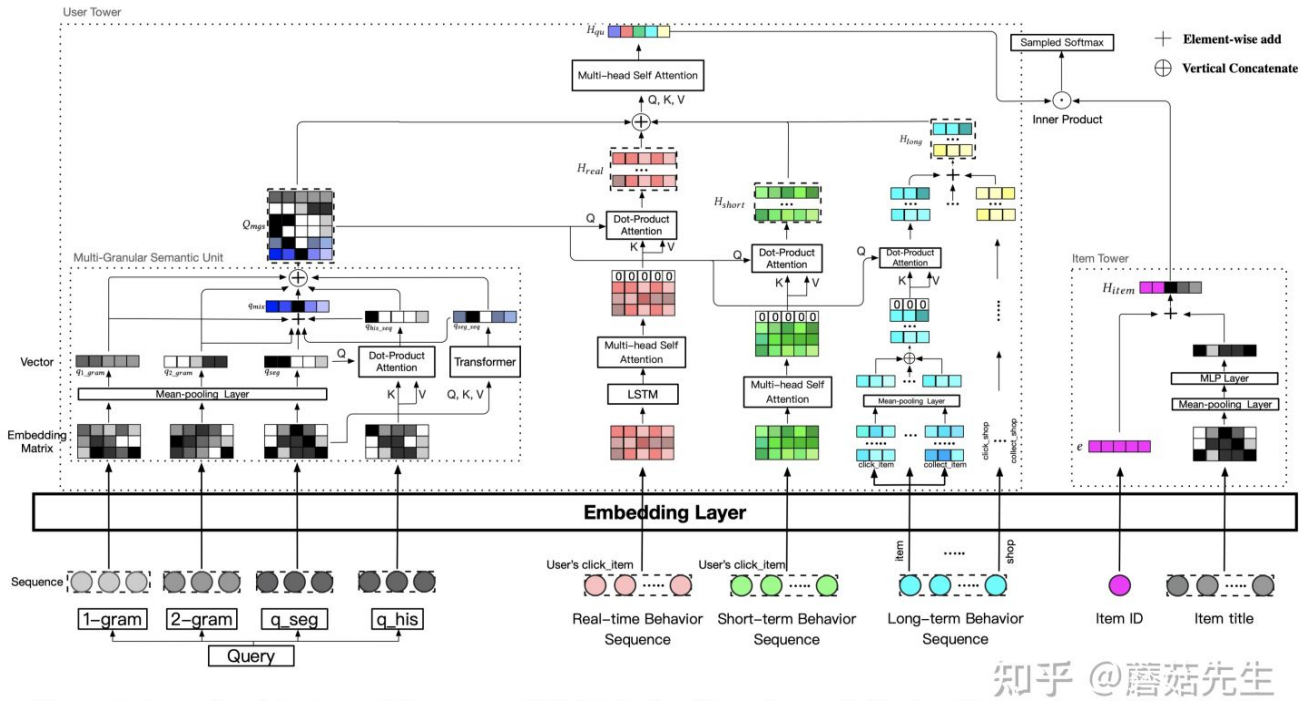

Figure 2: General architecture of the proposed Multi-Grained Deep Semantic Product Retrieval model (MGDSPR).

典型的**双塔结构**，在user tower部分做的比较重，item tower部分做的比较轻量。user tower输出用户embedding向量，item tower输出物品embedding向量，两者做点积得到预测值，再使用sampled softmax损失函数在全局item pool中进行优化。

2.1 User Tower

2.1.1 多粒度语义单元(Multi-Granular Semantic Unit)

淘宝搜索的query通常是中文。经过query分词后，每个分词结果的长度通常小于3。因此，作者提出了一种**【多粒度语义单元】**来多粒度地挖掘query语义。具体而言，输入：

- 当前query的**分词结果**$q_u = \{w_1^u, \ldots, w_n^u\}$，比如：{红色,连衣裙}
- 每个词w又由**字构成**，$w^u = \{c_1^u, \ldots, c_m^u\}$，比如：{红,色}
- 该用户的**历史搜索行为**$q_{his} = \{q_1^u, \ldots, q_k^u\} \in \mathbb{R}^{k \times d}$，比如：{绿色，半身裙，黄色，长裙}

可以获得如下6种粒度的表征：

- unigram：单字粒度的表征做mean-pooling，得到q_{1_gram}
- 2-gram：2-gram表征做mean-pooling，得到q_{2_gram}
- 分词粒度的词的表征做mean-pooling，得到q_{seg}
- 分词粒度的词表征看成序列，输入Transformer，再对最后1层隐层向量做mean pooling，得到$q_{\text{seg_seq}} \in \mathbb{R}^{1 \times d}$
- 历史搜索词q_{his}和当前搜索词的mean-pooling表征q_{seg}做attention加权融合（q_seg作为Query，q_his作为Key和Value）得到$q_{\text{his_seq}} \in \mathbb{R}^{1 \times d}$
- 混合表征：上述5种表征向量相加得到，得到$q_{mix} \in \mathbb{R}^{1 \times d}$

最终，$Q_{mgs} \in \mathbb{R}^{6 \times d}$由上述6种表征**concat**而成。

可以看到，作者从两个方面来充分地对query进行语义表征，**由此可以回答第一个问题**，query如何**充分地进行语义表征**?

- query**字面上的组织方式多样**：字粒度，2-gram粒度，词粒度。
- query的**表征方法多样**：pooling，transformer，concat，addition等。

当然，只讲结果，没有讲为什么这么做。有点过于经验性/实验性驱动，而不是问题/动机驱动。

2.1.2 用户行为注意力机制(User Behaviors Attention)

用户行为包括：用户的**实时、短期或长期**的点击或者购买行为。用户u在t时刻点击item i，用i_t^u来表示。对于物品i_t^u的表征向量，使用ID和side information(叶子类目、一级类目、品牌和所属店铺)做embedding。

和target-item注意力机制类似，此处使用query注意力机制来捕捉**用户历史行为**和**当前query**的**相关性。**目的是发现哪些**历史行为**和本次query**相关**，来**丰富用户在当前query下的语义/意图表征。**比如：历史购买行为，篮球鞋、裙子，此次搜索query是红裙，显然篮球鞋历史行为(可能送人的)对此次query毫无帮助，直接引入还会带来噪声，而裙子历史行为对此次query是有帮助的。

具体而言，在搜索场景中，用户的**历史行为**和**当前query**可能**都无关**，所以，作者加了一个**全零的向量**到用户的**行为数据**中，来消除**潜在噪声**和解决**用户历史行为和当前query可能完全无关**的情况。

个人认为这个优化点非常巧妙，如果不加全零向量，模型无论如何都会**强制关注到至少一个行为**，这在历史行为**和当前query都无关**的时候，显然是噪声。加了零向量后，**在完全无关**的时候，模型**attend到这个零向量**即可，**不会引入额外的噪声。**个人认为这个优化点在搜索场景中至关重要，也是和推荐场景差别较大的地方，鲜有论文会提到这点。

接下来介绍如何融合用户的**实时行为、短期行为和长期行为。**

- 实时**点击**行为序列：$\mathcal{R}^u = \{i_1^u, \ldots, i_t^u, \ldots, i_T^u\}$，其中，$i_t^u$是item embedding，是ID和side information embedding拼接在一起实现的。
- - 首先使用**LSTM**来捕捉用户行为的演变，得到LSTM的最后一层隐层输出，$\mathcal{R}_{lstm}^u = \{h_1^u, \ldots, h_t^u, \ldots, h_T^u\}$。

- 接着，使用multi-head self-attention来对$\mathcal{R}^u_{lstm}$的每个hidden state做聚合，得到$\mathcal{R}^u_{\text{self_att}} = \{h^u_1, \ldots, h^u_t, \ldots, h^u_T\}$。
- 接着，加一个全零的向量进去，得到：$\mathcal{R}^u_{\text{zero\_att}} = \{0, h^u_1, \ldots, h^u_t, \ldots, h^u_T\} \in \mathbb{R}^{(T+1)\times d}$ 。
- 最后，使用注意力机制，来获取和Q_{mgs}最相关的实时历史行为表征。具体的，拿组成Q_{mgs}的6个向量分别和实时行为序列做attention后（Q_mgs的向量作为Query，行为序列作为Key和Value），再拼接起来。

- 短期**点击**行为序列：$\mathcal{S}^u = \{i^u_1, \ldots, i^u_t, \ldots, i^u_T\}$。和实时行为序列表征相比，少了第一步LSTM，其它都一样。
- 长期点击/购买/收藏行为序列：

$$\mathcal{L}^u = \{i^u_1, \ldots, i^u_t, \ldots, i^u_T\}$$

由此可以回答开篇的第二个问题，query注意力机制而非target-item注意力机制以及引入零向量，能够保证捕捉和query相关的历史行为信息。

2.1.3 语义表征和个性化行为表征融合 (Fusion of Semantics and Personalization)

输入：

- 多粒度query语义表征：Q_{mgs}
- 个性化序列表征：$(H_{real}, H_{short}, H_{long})$

使用自注意力机制来捕捉二者的关系。特别的，作者添加了$[CLS]$ token在首位，形成输入：$I = \{[CLS], Q_{mgs}, H_{real}, H_{short}, H_{long}\}$。

输出：

然后将self自注意力机制的输出作为user tower的表征，

$$H_{qu} \in \mathbb{R}^{1\times d}$$

$$H_{qu} = \text{Self\_Att}^{first}([[CLS], Q_{mgs}, H_{real}, H_{short}, H_{long})$$

$$[CLS]$$

模仿BERT中的结构，可学习融合了query，实时、短期、长期用户序列的浓缩信息。

2.2 Item Tower

根据作者的实验经验，使用Item ID和Item的Title来获得Item的表征H_{item}。具体而言，给定item i的ID，其嵌入为：$e_i \in \mathbb{R}^{1\times d}$。给定title的分词结果$T_i = \{w^i_1, w^i_2, \ldots, w^i_N\}$，得到物品的表征，$H_{item} \in \mathbb{R}^{1\times d}$，即：

$$H_{item} = e + tanh(W_t \cdot \frac{\sum_{i=1}^{N} w_i}{N})$$

其中，W_t是可学习的变换矩阵。作者表示，通过实验发现，使用LSTM、Transformer等来捕捉title上下文感知的表征，其效果还不如上述简单的mean-pooling。给出的理由是：大部分的title由关键词堆叠而成，且缺乏语法结构。个人理解，可能想说字面上的语义信息足够凸显，**上下文信号较弱**，不需要复杂的模型来捕捉语义。

2.3 Loss Function

为了保证训练时的样本空间和在线推理时的样本空间一致，大部分工作会使用随机负采样的方法。但是这些工作都采用了pairwise hinge loss作为损失函数，只能进行局部的比较，和在线推理时需要的全局比较不一致。此外，hinge loss有一个非常重要的margin超参，其影响很大（当然sampled softmax不是也有温度系数超参吗？？）。为此，作者使用了softmax交叉熵损失函数，具体而言，给定正样本

$$i^+$$

，

$$\hat{y}(i^+|q_u) = \frac{\exp(\mathcal{F}(q_u, i^+))}{\sum_{i' \in I} \exp(\mathcal{F}(q_u, i'))}$$

$$L = -\sum_{i \in I} y_i \log(\hat{y}_i)$$

I是全部的item集合。实际上就是softmax交叉熵损失，然后因为I的数量很大，使用sampled softmax来优化即可（sampled softmax是full-softmax的无偏估计）。此处没有太大的创新点。在sampled softmax中，仍然需要负样本，参考京东的做法，作者使用**同一个batch内的其它样本**作为当前正样本i^+的负样本对，这个效果和使用随机任意的样本作为负样本差不多，而前者还能省不少计算资源。

接着，为了提高EBR系统的**相关性**，即增加更多相关性的样本进入后续的排序阶段。作者提出了两种优化策略，

- **对训练集中的样本进行噪声平滑**：作者引入了温度参数τ。此处也没有什么太大的创新点。τ无穷小时，相当于拟合one-hot分布，无限拉大正样本和负样本之间的差距；τ无穷大时，相当于拟合均匀分布，无视正样本还是负样本。作者认为，训练集中用户的点击和购买行为包含有不少噪声数据，不仅受到query-product相关性的影响，也受到图片、价格、用户偏好等诸多因素的影响，即用户点击/购买的item不一定和query相关，如果一味地拟合点击/购买行为，可能会带来很多相关性问题，因此引入温度参数来平滑，温度参数参数越大，则平滑的作用越明显，让模型不去过分关注点击样本，也花点"心思"去关注没有点击但是可能是相关的样本。文中取温度系数为2.形如：

  $$\hat{y}(i^+|q_u) = \frac{\exp(\mathcal{F}(q_u, i^+)/\tau)}{\sum_{i' \in I} \exp(\mathcal{F}(q_u, i')/\tau)}$$

- **生成相关性增强的困难负样本**：作者提出了一种在**embedding空间**自动生成困难负样本的方法。特别的，给定一个训练样本(q_u, i^+, i^-)，其中i^-是随机负采样的item embedding，q_u是user embedding，i^+是正样本item的embedding，为了得到困难负样本： 使用q_u去负样本中找到和其**点积最大**的top-N个负样本集合：I_{hard}，然后通过插值的方式，来混合正样本$i^+ \in \mathbb{R}^{1 \times d}$和困难负样本$I_{hard} \in \mathbb{R}^{N \times d}$，即：

  $$I_{mix} = \alpha i^+ + (1 - \alpha) I_{hard}$$

  $I_{mix} \in \mathbb{R}^{N \times d}$，形成N个困难负样本。其中，$\alpha \in \mathbb{R}^{N \times 1}$是从均匀分布$U(a, b)$中采样到的，$0 \leq a < b \leq 1$，显然，$\alpha$越接近1，生成的样本越接近正样本，即：生成的样本越困难。使用难负例来让模型区分正样本和正样本周围的这些负例(distinguish the positive sample i+ and its nearby

samples)。把生成的样本也纳入损失函数的计算：

$$\hat{y}(i^+|q_u) = \frac{\exp(\mathcal{F}(q_u, i^+)/\tau)}{\sum_{i' \in I \cup I_{mix}} \exp(\mathcal{F}(q_u, i')/\tau)}$$

- 可以通过调参a和b来控制负样本的"困难程度"。

由此可以回答开篇的第三个问题，通过引入温度参数进行噪声平滑以及生成困难负样本来保证EBR系统的相关性。

2.4 系统架构

最后，我们来欣赏下淘宝搜索引擎的系统架构。

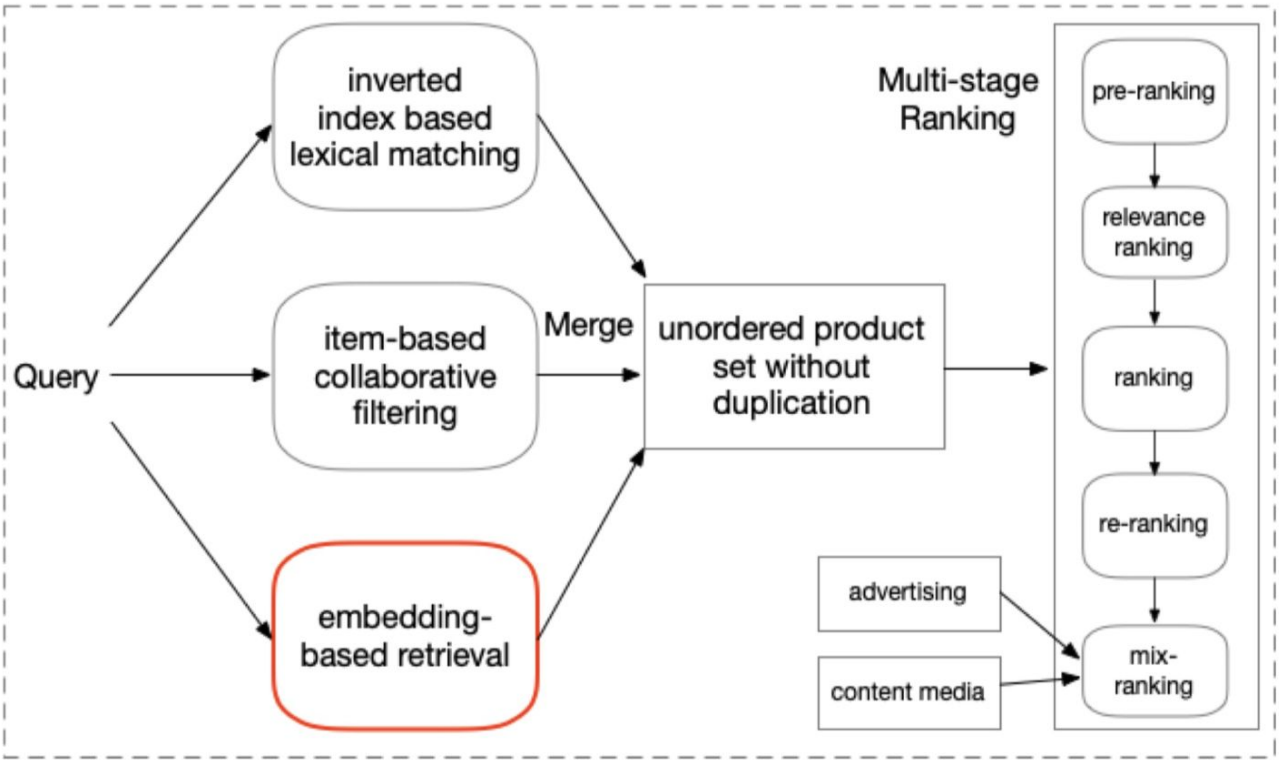

Figure 3: Overview of Taobao search engine.

搜索的整个过程如下：

- 用户发起一次请求
- 触发多通道检索系统，形成未排序的商品集合
- - 基于倒排索引的文本匹配
  - 基于Item的协同过滤（i2i）
  - 基于向量的检索
- 多阶段排序
- - 粗排
  - 相关性排序（removing products that are inconsistent with the predictions of the query's category）
  - 精排
  - 重排

- 混排：商品、广告、多模态内容

本文重点在基于向量的检索：

- **离线**：使用分布式Tensorflow对过去**1周内**的搜索日志数据进行训练，上线后**天级更新**模型参数。
- **部署**：item tower离线算好所有product的向量，并存在ANN索引系统里，product量级巨大，分片存，共6列，借助层次聚类算法做量化降低存储开销(实际上猜测就是Faiss)；query/user network做实时serving。实际检索的时候，能够实现类似布尔检索系统的高效检索。
- **性能**：实时检索9600个item,从6列的离线索引中分别召回9600 / 6 个最相近的item。98%的item在10ms内能检索完，即：98线为10ms。很强的性能了。

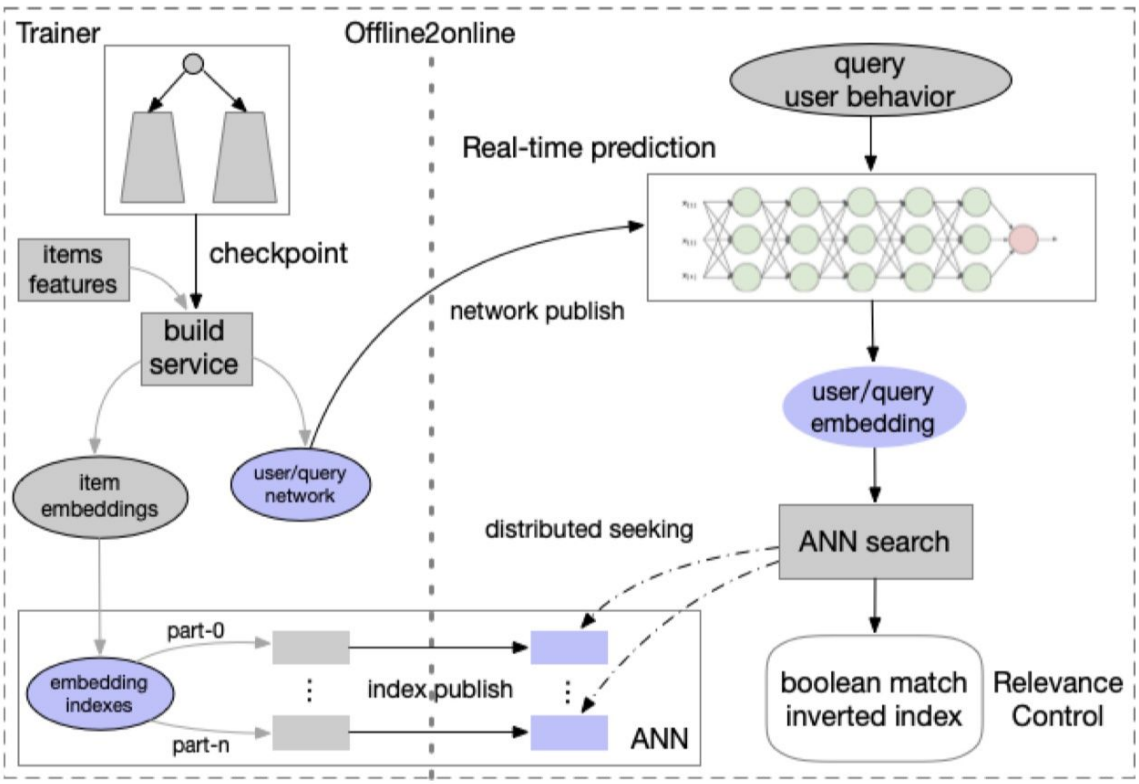

Figure 4: Deployment system of our MGDSPR model.

还有个很重要的相关性模块还没有介绍。开篇提到过，EBR检索系统在**个性化和模糊匹配方面做的很好**，但是相关性上缺点也很大。归根结底在于，EBR不是exact match策略。而结构化检索中，品牌、颜色、类型等结构化字段能够很大程度上保证相关性。但是EBR却做不到这点。

比如：用户检索阿迪达斯运动鞋，那么完全匹配查询能够去检索品牌：阿迪达斯，类目：运动鞋；但是EBR可能在embedding空间检索到耐克运动鞋，这显然是不相关的，会影响用户的体验。

因此，作者在ANN结果的后面，又加了层基于boolean matching的相关性控制模块。首先对query进行了**query理解**，识别出品牌、类目等意图，然后对item的title中也挖掘出品牌、类目等**结构化字段**，然后用这些查询理解的意图过滤掉未命中这些结构化字段取值的item。

作者还提到，Facebook的文章是通过EBR系统来弥补基于完全匹配的检索系统在个性化、模糊匹配上的不足；而淘宝搜索相关性控制模块的出发点相反，是通过基于完全匹配的检索系统来提升EBR系统的**相关性**。总之，二者相辅相成。

3. Evaluation

离线实验以及实现细节也是工业界文章的核心亮点，值得大家好好品。

3.1 Settings

- **离线指标**:
- - $\mathrm{Recall@K}$。用户点击或者购买的item作为ground truth，召回出的Top-K Item作为我们预测的结果。作者提到，在检索阶段，用AUC做离线指标时，和线上的GMV指标无法保持一致，而召回指标则可以。
  - P_{good}。相关性指标，即Top-K结果中有多少个结果和query强相关，即：相关item的数量比例。是否相关的label不是人工标注的，而是采用了一个训得很好的相关性模型(在单独的人工标注数据集上的AUC能够达到0.915)来打标。
  - Num_{prank}, Num_{rank}。衡量EBR检索系统对各个阶段的影响指标。即：这一路召回的Top-K Item中，有多少会进入后续的各个排序环节，进入越多，说明相关性保证的越好。
- **在线指标**:
- - GMV = #pay amount.
  - 用户搜索体验指标：P_{good}和$P_{\mathrm{h_good}}$表示展示给用户的item中，和query相关的占比。前者用模型打标，后者外包标注。
- **实现细节**:
- - **网络结构：**
  - - 实时行为序列最大长度50，长短期行为序列最大长度都是100。对于那些比最大长度要小的行为序列样本，需要使用带mask机制的attention，使真实的last hidden state只能关注到前面的序列。
    - user tower, item tower, LSTM等结构的隐层维度数为128。
    - 实时行为中，LSTM结构2层，dropout=0.2，LSTM之间加残差连接，自注意力机制头的数量是8。
  - **训练：**
  - - batch大小为256。
    - 困难负样本中，均匀分布a和b的值为0.4，0.6；生成的困难负样本数量为684。
    - 温度参数$\tau = 2$。
    - 随机初始化所有参数。
    - AdaGrad，初始学习率0.1。
    - 梯度裁剪，当梯度的L2范数超过3时做裁剪。
  - **配置：**
  - - 分布式机器学习平台，20个PS，100个GPU作为worker，配置是Tesla P100。
    - 训练时间：3500万步，耗时54小时。
- **数据集，** 淘宝真实的搜索行为数据集，2020年12月连续8天的点击和购买日志。
- - **训练集**：前7天作为训练集，约47亿条
  - **测试集**：从第8天中，随机从搜索系统数据中抽100W条query数据；从推荐系统数据中抽50W条购买数据。

全量的候选item的数量级是1亿，和线上真实推断时的候选集保持一致。

3.2 离线对比实验

- Baseline：α-DNN，MLP结构，很强的baseline。静态特征，统计特征，序列特征做pooling作为输入。
- MGDSPR：本文的方法，如上文所述。作者提到一点，加入统计特征到MGDSPR中，recall指标并没有提升。挺神奇的。可能行为序列信息捕捉地足够好，不需要过多统计特征。

Methods	Recall@1000	P_{good}	P_{f_good}	Num_{prank}
a-DNN [5]	82.6%	70.6%	83.2%	769
MGDSPR	84.7%(+2.5%)	80.0%(+13.3%)	84.1%(+1.1%)	815(+6.0%)

提升还是挺大的，尤其是相关性样本的占比，以及进入粗排的相关性item的数量。说明EBR提前考虑了相关性后，在相关性控制模块中不会被过滤太多不相关的ITEM，有更多的相关ITEM进入后续的排序环节。

3.3 消融实验

- mgs：2.1.1中提出的多粒度语义单元，对recall和相关性指标都有用。
- trm：2.1.3中的语义表征和个性化行为表征做融合，对recall和相关性指标都有用。
- τ： 2.3中的温度参数。对召回指标负向，但是对相关性指标提升非常显著。
- I_{mix}，对召回指标负向，对相关性指标帮助大。

Methods	Recall@1000	P_{good}
MGDSPR	85.6%	71.2%
MGDSPR + mgs	86.0%	71.6%
MGDSPR + trm	86.4%	71.4%
MGDSPR + τ	85.5%	79.0%
MGDSPR + mgs + trm + τ	86.8%	79.2%
MGDSPR + I_{mix}	83.6%	75.6%
MGDSPR + all	84.7%	80.0%

看了消融实验，对召回指标帮助大的是mgs和trm；对相关性指标帮助大的是温度参数和困难负样本。

3.4 Online A/B Test

Launched Platform	GMV	#Transactions
Taobao Search on Mobile	+0.77%	+0.33%

有挺大的线上指标提升。

其它的分析实验比较常规，总结下来就是：

- softmax收敛速度比pairwise loss快，recall指标也高不少。
- 温度参数和困难负样本对相关性的提升很大。

4.Summarization

总体而言，这篇文章干货很多，细读会发现很多细节。有几大亮点，

- **多粒度语义单元**，对query语义的多粒度挖掘和表征，值得在搜索场景中尝试。
- **用户行为序列在搜索场景中的建模方法**，query attentive而不是target-item attentive以及零向量的引入是最大亮点，长短期融合等也值得实践。
- **EBR系统对相关性的保证**，softmax损失函数+温度参数；在embedding空间生成困难负样本。