电商搜索：召回篇

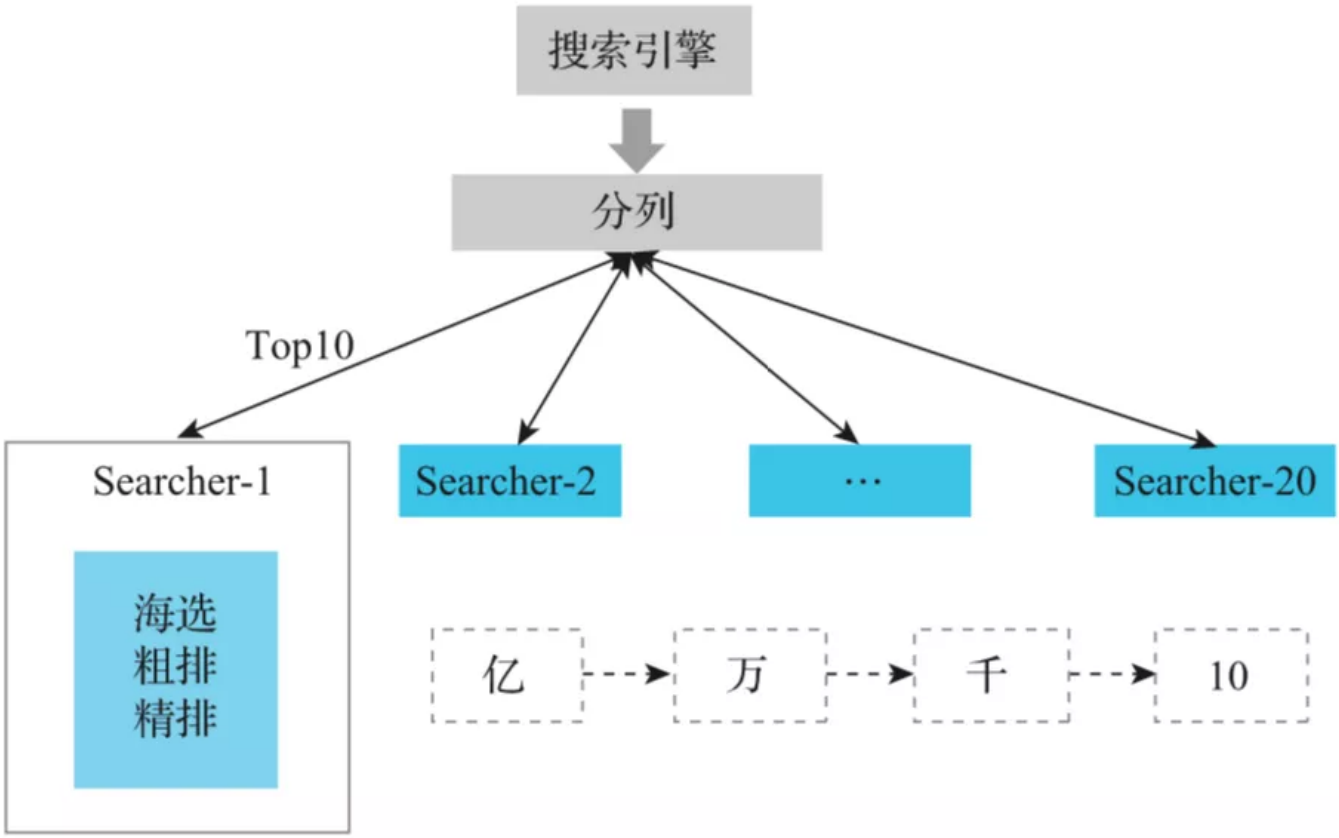
阿里CBU技术部 AI启蒙者 1周前

文章作者：阿里CBU技术部

内容来源：《阿里巴巴B2B电商算法实战》

导读：搜索引擎已经广泛应用于电商网站，我们为什么需要搜索引擎?搜索引擎又是如何工作的呢?电商网站中存在着上亿的商品，想象一下，当我们想要买一 件T恤的时候，可能会在页面上搜索“圆领 T 恤”，最直观的想法是遍历所有商品，查找满足需求的商品，这样的做法效率低下，可以采取与数据库相似的处理方法，可以通过构建索引的方式来加速对商品的检索。这里需要引入倒排索引的概念。对每个商品需要被检索的子单元创建一个索引，在搜索中可以直接通过子单元快速找到商品，而不需要遍历所有商品，这个索引就是搜索引擎中的倒排索引。

利用倒排索引对商品进行初筛的过程，我们称之为召回阶段。在这个阶段，不会进行复杂的计算，主要根据当前的搜索条件进行商品候选集的快速圈定。在此之后，再进行粗排和精排，计算的复杂程度逐渐提高，计算的商品集合逐渐减小，最终完成整体的计算。



搜索引擎工作流程

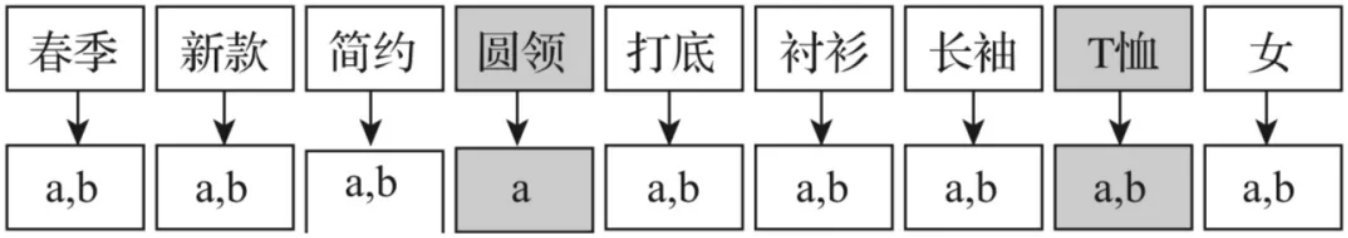
上图所示框架为搜索引擎工作的主要流程。在引擎的构建过程中，首先进行分列，在每个分列中单独完成召回、粗排和精排的过程，最终从每列中选出分数最高的 N 个商品，并在顶层进行聚合，选出全局分数最高的 N 个商品，即可完成整体的排序过程。

01

词召回

在构建引擎索引的过程中，首先要解决的是分词问题。我们仍以“圆领 T 恤”为例，目前存在一个标题为“春季新款简约圆领打底衬衫长袖 T 恤女”的商品 a，存在一个标题为“春季新款简约打底衬衫长袖 T 恤女”的商品 b，我们首先选择合适的分词粒度来对这两款商品进行分词，并将分词项构建为倒排索引的词典。需要指出的是，分词粒度并不是越小越好，如“简约”，如果拆分为“简”“约”两个单元，则无法真实传达词语的含义。我们按照小粒度分词的结果建立索引，对于搜索词“圆领 T 恤”，同样先进行分词，分词结果为：

建立的倒排索引可以表示为：



这里我们可以先将搜索词改写为“圆领”AND“T恤”的查询语法树。在进行召回时，通过“圆领”可以召回商品 a，通过“T 恤”可以召回商品 a 和 b，取交集，则可以召回到合适的商品 a。

以这种倒排索引为基础，我们不仅可以对商品的标题进行分词并构建索引，也可以对商品的类目、价格、是否包邮等内容建立相应的索引，只需要构造正确的查询语法树，即可快速召回符合多种组合条件的商品。

即使允许召回更多的商品，仍旧有召回数量的限制，所以我们需要给每个商品一个可以用于召回的分数，进行召回部分的截断。召回的最终目标是召回在精排阶段分数尽可能高的商品集合。由于这个分数将被用来构建倒排索引，每天进行一次，每个商品只有一个分数，所以这个分数无法与用户和搜索词进行关联，所以我们主要考虑商品质量和发布时间。最简单的方式是直接使用精排的重要分数进行拟合，也可以依据线上的情况进行召回分数的学习，学习的目标是使 TopN 的集合与精排产生的集合重合度更高。

加入商品的召回分数之后，仍然会存在相关性及类目倾斜的情况。举个简单的例子，当我们搜索“手机”的时候，同时可以命中非常多的手机配件。手机配件的商品量远远多于手机，它们价格更低，重复采购的周期更短，在召回分数上，很容易超过手机产品，极端情况下，可能出现手机产品召回较少甚至无法召回的情况。为了缓解这类问题，搜索引擎也支持按照字段限制召回数量的功能，我们可以结合类目预测的结果，对每个搜索词计算每个类目下最合适的召回数量，从侧面保证搜索结果的类目相关性与多样性。

02

向量召回

前面介绍了词召回的相关内容，而在搜索引擎和搜索广告中，还有一种很重要的问题是召回的时候保持语义相似度，这里主要体现在两个方面：召回和排序。在召回时，传统的文本相似性算法如TF-IDF、simhash、BM25 等，很难召回具有语义相似性的 Query-Doc 结果对，如“儿童用品”与“玩具”的相似性、“数码产品”与“耳机”的相似性、“微软 Surface”与“平板电脑”的相似性。

与传统的文本类似，基于词的召回也会存在无法自动扩展词的语义信息的问题。例如，当用户搜索一个数码产品的时候，如果商家的产品词没有加上“数码产品” 这个词，那么他的产品就无法被倒排索引正确召回。为了解决这个问题，一方面，可以利用近义词对用户查询词进行扩展之后再进行召回，增加被正确召回的商品数量；另一方面，还可以通过深度学习召回模型解决这个问题。首先，我们介绍召回问题的数学定义。

召回问题可以定义成分类问题，输入样本集为 $\boldsymbol{x}_i$、 y_i，其中 $\boldsymbol{x}_i$为样本，y_i为召回的类别，那么在这个定义中，损失函数为交叉熵损失函数(Cross Entropy Loss)：

$$L(\hat{y}, y) = -\sum^{C} y_i \log(\hat{y}_i)$$

其中y是one-hot编码，只有对应的类别标签才为1，$\hat{y}$ 是每个类别输出的概率。

召回问题也可以定义成：输入一个查询样本 $\boldsymbol{x}_q$，从数据库中召回与查询样本相似的样本$\boldsymbol{x}_p$，形成样本对（ $\boldsymbol{x}_q$，$\boldsymbol{x}_p$），根据召回结果是否相似，可以得到多个正负样本对。在这种样本对的定义下，有两个损失函数定义方式：样本对的损失（ Pairwise Ranking Loss ）和三元组的损失（ Triplet Loss)。

样本对的损失如下（ Positive Pair 和 Negative Pair 分别表示正样本对和负样本对)：

$$d(\boldsymbol{x}_q, \boldsymbol{x}_p) \text{ if positive pair}$$

$$\max(0, m - d(\boldsymbol{x}_q, \boldsymbol{x}_p)) \text{ if negative pair}$$

在这个定义下，总体的损失函数如下：

$$L(X_1, X_2, Y) = \sum \frac{1}{2} d(\boldsymbol{x}_q, \boldsymbol{x}_p)^2 + \frac{1}{2} (\max(0, m - d(\boldsymbol{x}_q, \boldsymbol{x}_p)))^2$$

如果把每个样本都表示成三元组的形式，那么损失函数如下：

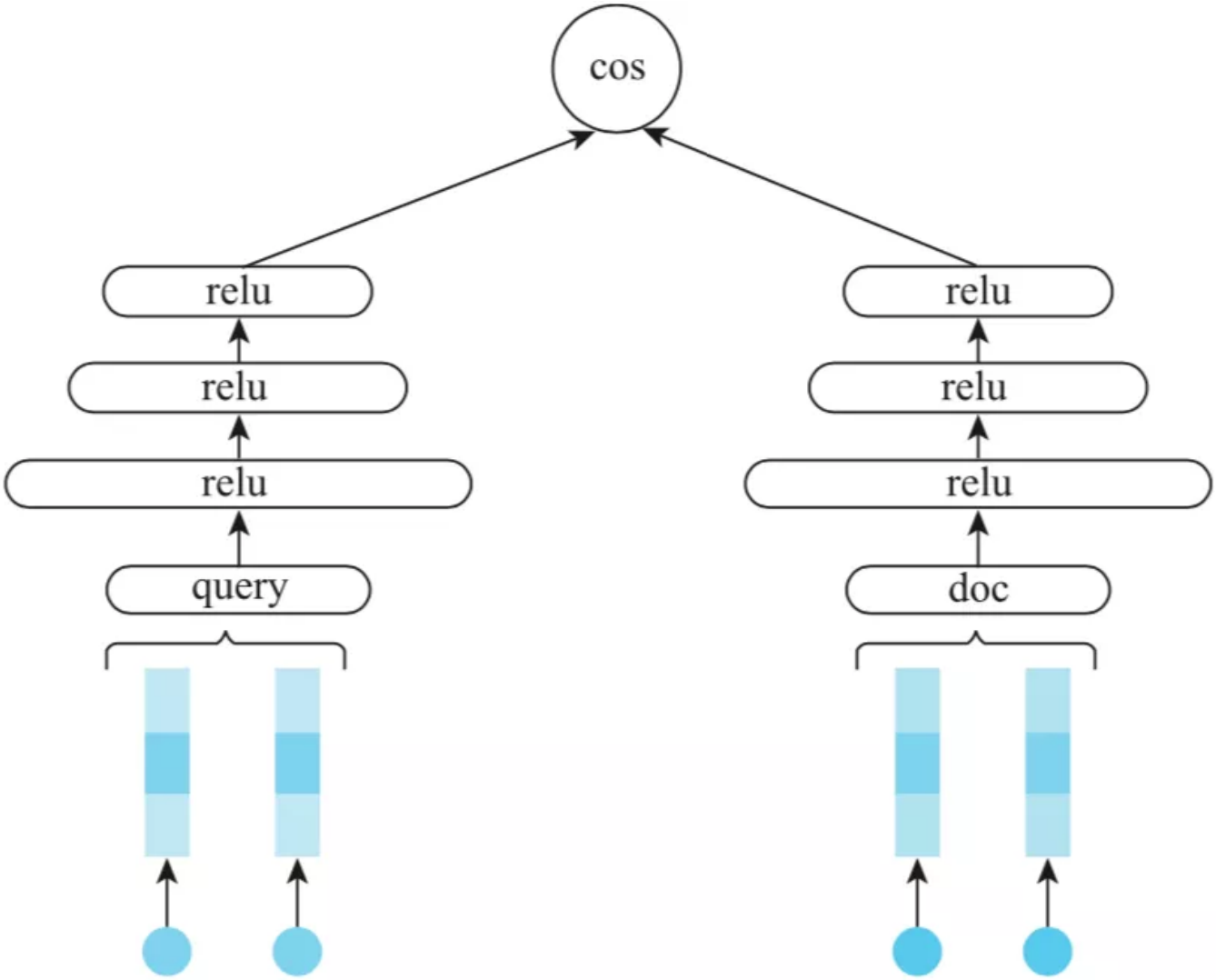
$$L(X_q, X_p, X_n) = \sum \max(0, d(\boldsymbol{x}_q, \boldsymbol{x}_p) - d(\boldsymbol{x}_q, \boldsymbol{x}_n) + m)$$

前面提到，传统的词召回方法无法有效解决语义相似度问题，为了解决这个问题，我们在搜索排序中引入了基于向量召回的多路召回方法对词召回方法进行补充，并实现了个性化实时等向量召回模型。下面简单介绍这些模型。

1. 基于深度学习的召回模型

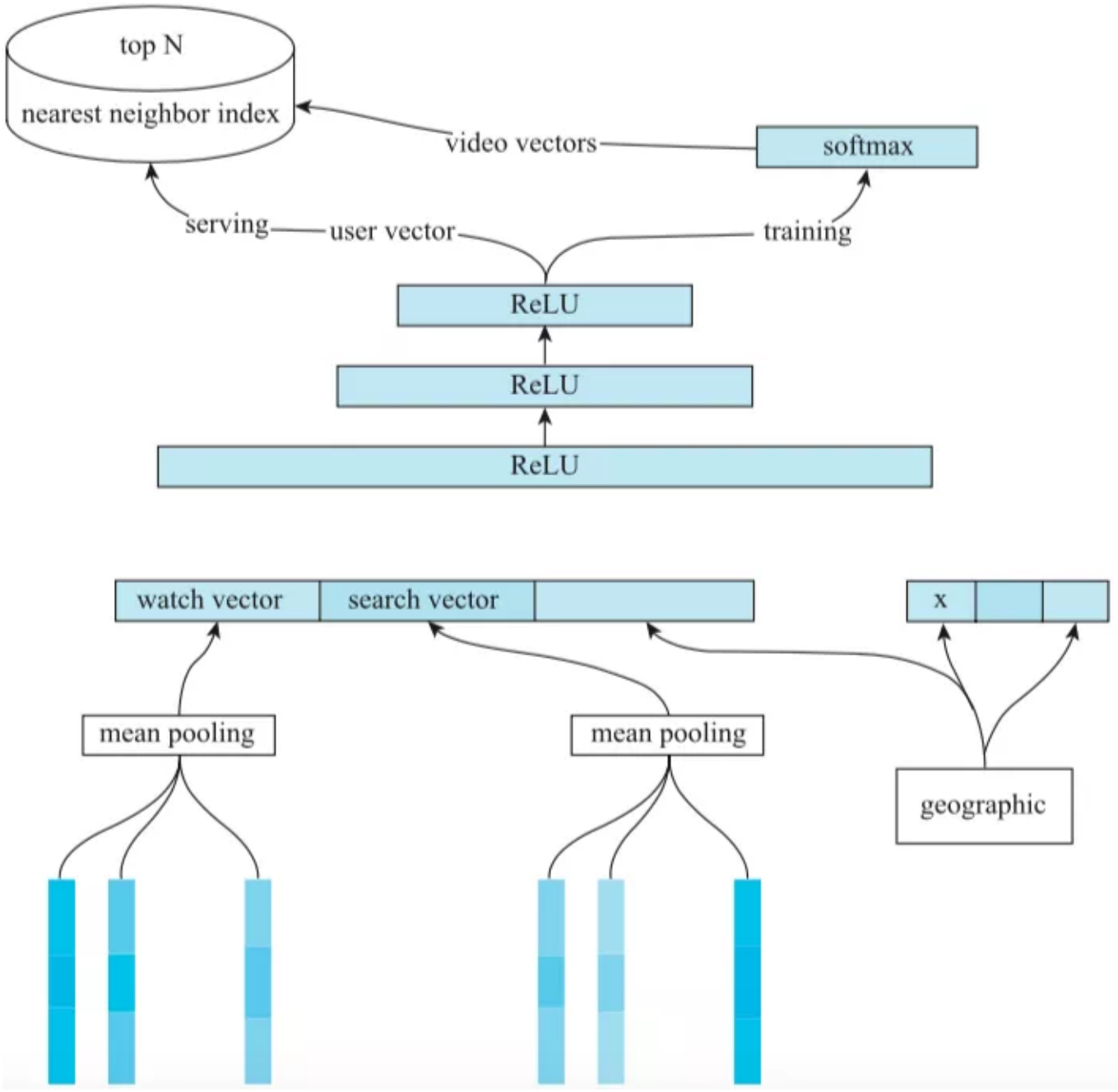
在搜索推荐领域中，使用分类的方法召回模型大致有两种思路，分别是微软提出的 DSSM 方案和谷歌提出的 DeepMatch 方案。

2013 年，微软提出了 DSSM 模型（Deep Structured Semantic Model），其结构如下图所示，核心思想比较简单，使用了搜索引擎中的曝光点击数据，利用几层的 DNN 把查询词及查询文档 Embedding 成语义向量，并通过约束两个语义向量的余弦距离，最终训练出语义相似度模型。该模型可以直接学习出查询词及查询文档的 Embedding 向量，用于计算查询词与查询文档之间的语义相似度。



DSSM 模型基本结构

2016 年，谷歌发表了应用在 YouTube 视频网站的推荐算法，据这篇论文介绍，他们的视频推荐分为两个阶段，第一阶段是候选集产生（Candidate Generation），其核心思想是通过分类的方法做召回。Serving 阶段使用向量召回 TopN 的视频作为候选集，并进入第二阶段—视频排序（Ranking），最终给用户推荐他感兴趣的商品。在这里，主要关注的是第一阶段的方案，使用分类的方式完成商品的召回。其基本结构如下图所示，模型的输入包括用户最近观看的视频序列、搜索词序列，以及其他的统计特征。通过几层的 DNN 网络预测出用户下一次会点击观看的视频，我们把这个模型称为 DeepMatch。



Deep Match 模型基本结构

前面提到，DeepMatch 模型的预测目标是给定多个商品，预测出用户最可能点 击的那个商品。因此，基于多分类的思想，可以得到 softmax 函数：

$$P(w_t = i \mid U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_i u}}$$

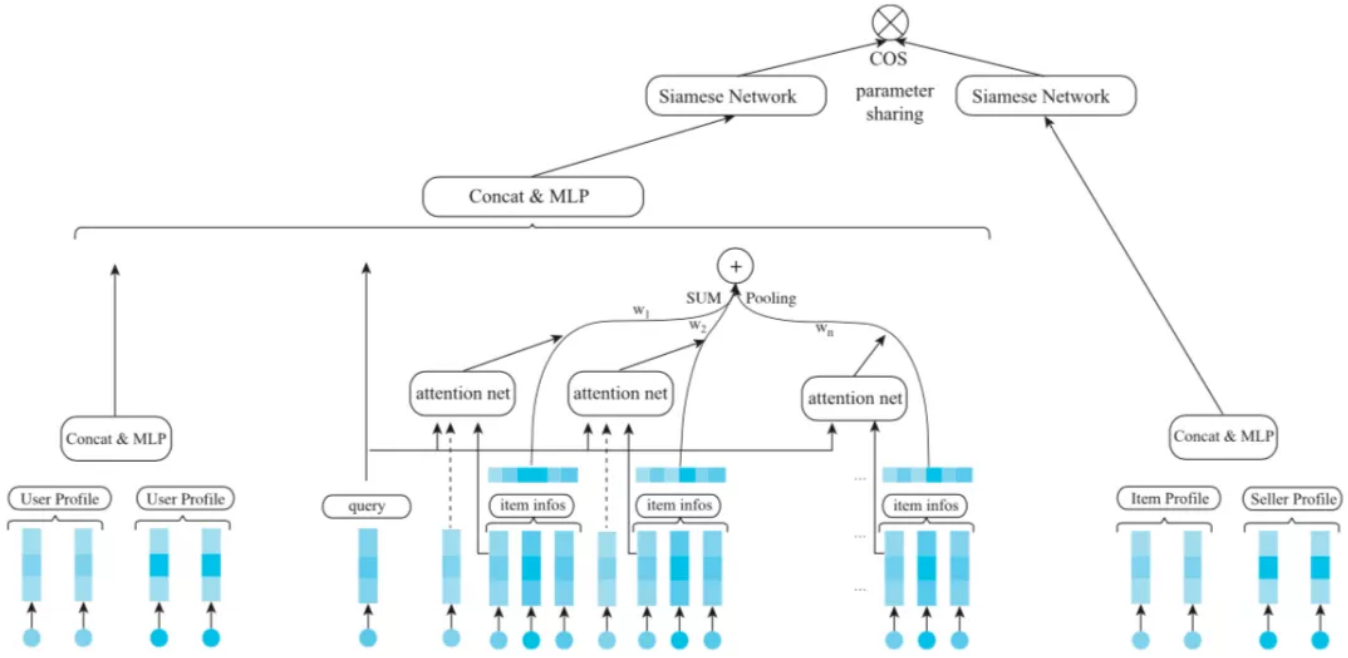
这里的 u 代表用户的 vector 表征向量，v_i 实际代表了第 i 个类别的参数 vector，原论文里一个类别就是一个商品。

这个模型在线上使用的时候，可以直接保存每个商品的参数 vector，这样，在实际应用的时候，只需要实时预测出用户的向量 u，就可以通过向量的相似度快速召回 TopN 的商品。

2. 用户实时个性化召回模型

在 1688 的向量化召回方案中，我们借鉴了 DSSM 模型和 DeepMatch 模型，基于用户的曝光点击日志构造了基于深度学习的召回模型。下面简要介绍我们使用的 match 模型。

1688 网站中，用户查询的 Query 词语一般比较简短，这种情况下，满足 Query 召回条件的商品会非常多，为了满足搜索效率，需要尽量保证召回商品的有效性。这里的有效性定义为符合用户的个性化需求，使得用户更有可能去点击这个商品。由于全站商品数量巨大，直接使用全量的商品作为分类向量会导致模型的参数量非常大，而且为了接入商品的 side information 辅助信息，我们采用了 DSSM 方案。具体采用的模型结构如下图所示。

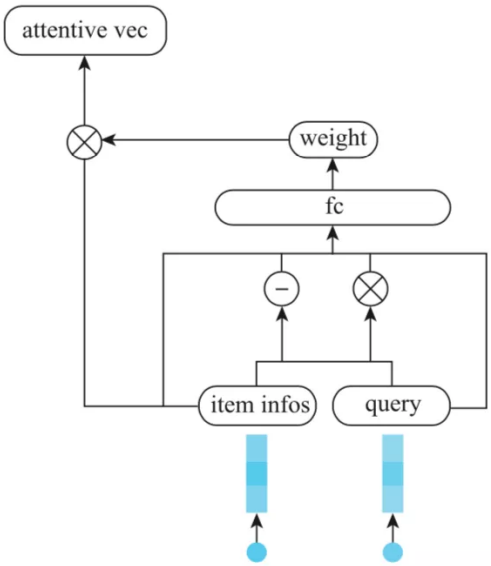


个性化向量召回模型结构

电商网站中，用户的行为序列一般是比较丰富的信息，也是很重要的信息，特别是用户的实时行为序列，比较明显地代表了用户的当前兴趣，有助于识别出用户下一个会点击商品。召回模型分为两个部分，上图左边部分负责学习用户在当前查询下的 user-query-vec 向量，这个用户查询向量是实时更新的。模型的输入主要是用户的人口统计学特征及实时行为序列。上图右边的部分负责学习商品的向量。

为了捕捉用户的实时兴趣，我们对用户的实时行为序列进行了建模，模型利用了用户的 Query 词对行为序列上面的所有商品进行 Attention 学习，同时为了突出每个行为类型及行为发生时间的不同权重，再基于 context 特征对行为序列进行建模。具体体现为引入了一个分层的 Attention 网络对模型进行建模。

① 基于 Query 与 item_infos 进行 Attention 计算



查询词与商品的权重计算

如图所示，假设 Query 的 vec 为 $\boldsymbol{q}\in R^n$，第 i 个商品信息组合成的 vec 为 $f_i \in R^n$，采用类似于 ESMM的方式连接两个 vec，如下：

$$\hat{\boldsymbol{v}}_i = [\boldsymbol{q} - \boldsymbol{f}_i, \boldsymbol{q} \odot \boldsymbol{f}_i, \boldsymbol{q}, \boldsymbol{f}_i]$$
$$\hat{\alpha}_i = \boldsymbol{W}_k \sigma(\boldsymbol{W}_{k-1}\sigma(\cdots \boldsymbol{W}_1 \hat{\boldsymbol{v}}_i + \boldsymbol{b}_1)\cdots) + \boldsymbol{b}_{k-1}) + \boldsymbol{b}_k$$

其中 σ 为非线性激活函数，一般采用 ReLU。

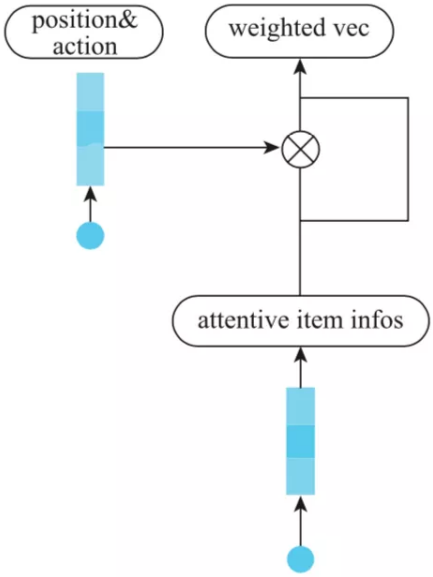
多个商品再使用 softmax 函数对权重归一化，如下：

$$\alpha_i = \frac{\exp(\hat{\alpha}_i)}{\sum\{\exp(\hat{\alpha}_i)\}}$$
$$v_i = \alpha_i * f_i$$

② 基于 context 与 item_infos 进行 Attention 计算

基于 Query 与 item_infos 的 Attention 网络，计算得到了一个 attentive item vec。

搜索推荐中，用户最近的浏览行为对下一个浏览的商品有着很重要的影响，因为用户在一个 session 下，浏览的商品具有很高的相似性，而且越近的行为一般影响会越大。因此，商品序列的序号可以作为描述行为重要性的特征，同时我们还考虑每个商品的下一步行为。例如，一个商品被浏览之后，还被用户收藏，或者被加入购物车，那么这个商品对用户的下一个浏览行为的影响就很大。同样会产生影响的还有用户停留时间等行为。我们参考了谷歌 2018 年提出的论文，引入了 latent cross 的方式对行为序列进行建模，把这些描述每个行为的特征组合起来，作为行为序列的 context 特征。



基于上下文特征的商品重要性建模

如图所示，假设经过第一层 Attention 的 item 向量为 v_j，行为序列的 context 特征向量为 c_j，那么基于 latent cross 的 Attention 计算如下：

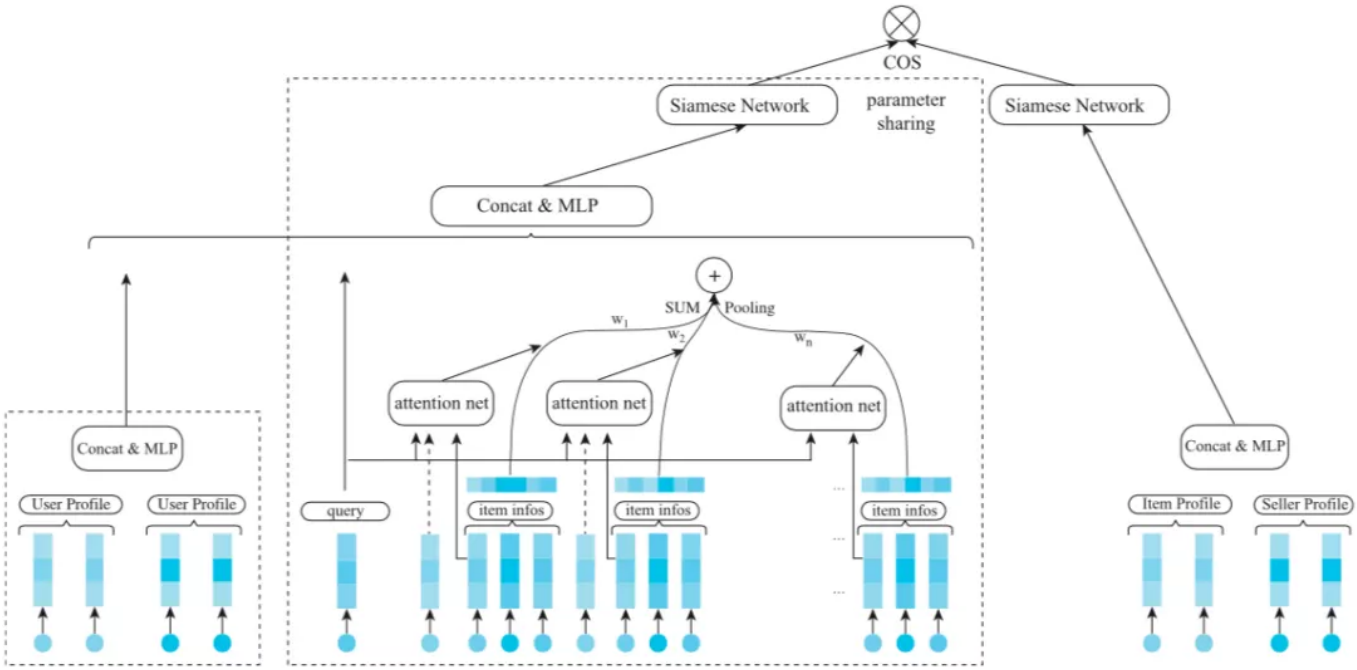
$$\overline{v}_i = (1 + c_i) \cdot v_i$$

最终得到 Attention 之后的商品 vec，用商品的 vec 的 sum pooling（即$\sum \overline{v}_i$）代表用户的实时兴趣。

学习出来的用户行为序列向量与其他的特征学习出的向量，结合查询词的 Embedding 结果共同 concat 起来，最终得到用户的向量表达。对于商品的向量，模 型也采用了 DNN 结构对其进行

学习。除了输入用户行为向量以外，还会输出一些id 类、统计类及商品效率类特征，例如点击率、转化率等。如果对 id 类特征进行了 Embedding，dense 特征就会作为 raw 特征输入到模型中。

我们的损失函数比较简单，使用了交叉熵作为模型的损失函数。采样本的时候使用了当前 Query 下用户点击的商品作为正样本，并从点击的位置附近采集部分样本作为负样本，同时加入了同类目下其他未点击的样本作为负样本，按照样本的历史点击率对样本进行重要性采样。在这个召回方案中，我们没有采用 Pairwise Ranking Loss 或者是 Triplet Loss，因为采用这两种方式的建模会增大样本数量，延长模型的训练周期。后续可以考虑进一步优化。



线上模型分解示意图

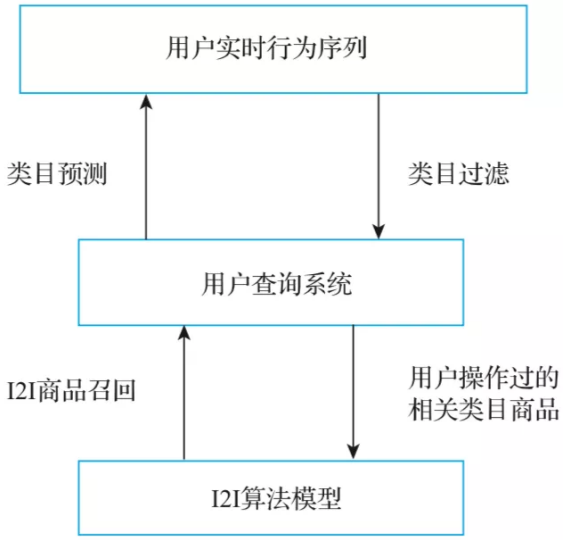
线上部署的时候，为了加速计算的性能，我们把模型分成几个部分进行计算，如上图所示。把用户向量网络分解成两个部分，图中左边部分是一个离线预先算好的向量，即把用户的 id 类特征以及其他的统计类特征在离线的时候预测成一个向量；右边是一个实时计算模块，流式计算平台负责收集用户的实时行为，当用户发起一次查询时，把用户的实时行为序列以及左边部分学习出来的向量作为右边网络的输入，最终得到用户的实时向量。同样地，商品的特征也可以离线预先计算好。线上召回的时候，使用用户实时向量快速召回 TopN 个商品向量。

3. 多路召回

除了线上的实时用户个性化召回模型，我们还实现了其他的召回方式，下面也简单介绍一下。

① 基于用户历史行为序列的 I2I 召回

如图所示，从用户操作过（例如点击或者收藏）的商品中获取符合用户当前查询类目下的所有商品，并对这些商品进行 I2I 扩展，得到更多用户感兴趣的商 品，从而召回更多有效的商品。



基于用户行为商品的 I2I 召回

② 查询词与商品文案，标题信息召回

对于 1688 网站，智能运营算法团队会为所有的商品生成关键文案，文案会突出商品的一些特征，例如商品的纹理、材质、颜色、营销信息等。结合商品的标题，我们基于 DSSM 结构实现了基于查询词与商品标题、商品文案信息的召回方案。

我们统计了用户的 Query 词的数量及 title 关键词的数量分布，发现 Query 词的长度偏短，而 title 关键词的数量比较多。绝大部分查询词集中在 1~4 之间，因此我们采用 meanpooling 方式表达 Query 词。但是对于 title 关键词，我们使用 self-Attention 的方式对商品的 title 及文案信息进行建模，让模型学习出 title 及文案信息之间的组合方式。

通过多路召回的方案丰富了商品的数量，对基于文字匹配召回不足的查询词起到了很好的补充作用。

今天的分享就到这里，谢谢大家。