搜索系统算法解析-OAA

向量召回

当前搜索业内主流和主力的召回通路无疑是向量召回，相较于传统的关键词匹配，向量检索能够捕捉语义相似性，使其在应对模糊查询时也能有不错的检索效果，比传统的基于关键词的检索方法更具容错性。此外，向量表征可以高效融合多模态、个性化等更丰富的信息，对于提升搜索引擎的准确性和用户体验至关重要。

基于Embedding的向量检索（EBR）是一种使用向量表征查询Query和文档Doc，并将检索问题转换为向量空间中的 **近似最近邻搜索** （Approximate Nearest Neighbor，ANN）搜索问题的技术。其基本步骤如下：

1. **向量表示：** 训练模型，将查询和文档映射到向量空间中
2. **建立索引：** 离线将海量文档集合转化为向量，同时为每个文档建立索引
3. **近似检索：** 将查询经过向量化处理，利用查询向量与所有文档向量之间的相似度进行计算实现检索，即查找与查询向量最接近的TopN个文档向量，文档向量对应的文档作为召回结果（由于在大规模数据集上计算全量的相似度非常耗时，通常会使用近似最近邻算法来加速检索过程）

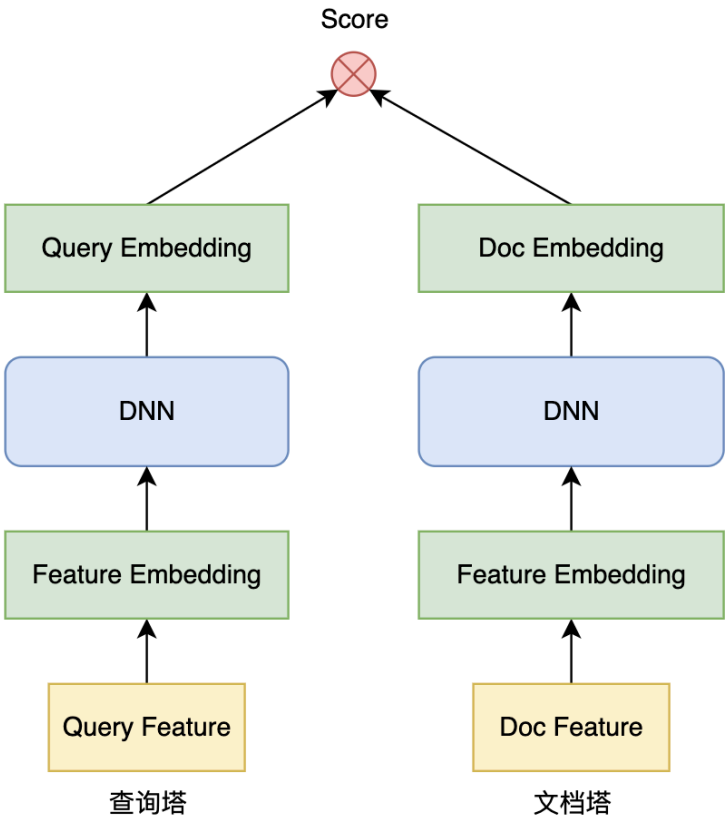
对于向量召回的技术框架可以统一为以下四个方向展开：**设计模型结构**、**构建训练样本**、**定义优化目标**以及**线上推理和模型更新**。

模型结构设计

向量召回采用双塔模型结构（应对数以亿级的海量文档），双塔模型由 **查询塔（Query Tower）** 和 **文档塔（Doc Tower）** 组成，这两部分模型通过共享或非共享的方式对查询和文档分别进行编码，最终将它们映射为固定维度的向量表示。

当查询和文档映射到向量空间后，通过 **余弦相似度（Cosine Similarity）** 或 **内积（Dot Product）** 计算两者的相似度。模型训练时，优化目标是最大化相关查询-文档对的相似度，同时最小化不相关查询-文档对的相似度。

通常来说，使用 Cosine 比 内积 效果要好，Cosine 相似度通过归一化消除了向量长度的影响，即更加关注向量的**方向**（角度），而内积则受向量长度的影响。另外，采用内积计算相似度时，对向量进行 **L2归一化** 后再内积等价于 Cosine。



查询塔

模型输入

查询塔的模型输入特征通常有三个维度：Query特征、User特征、User-Query交叉特征：

特征	类型
Query文本	Query特征
Query属性特征（类目/NER/主题/意图等）	
Query搜索统计特征（周期内的曝光量/点击量等）	
User属性特征（User ID/性别/年龄/所在省市/画像）	User特征
User历史搜索词/点击文档序列	
User历史互动行为特征（周期内的搜索/点赞/收藏/评论等统计量）	
Query-User交叉特征	Query-User交叉特征

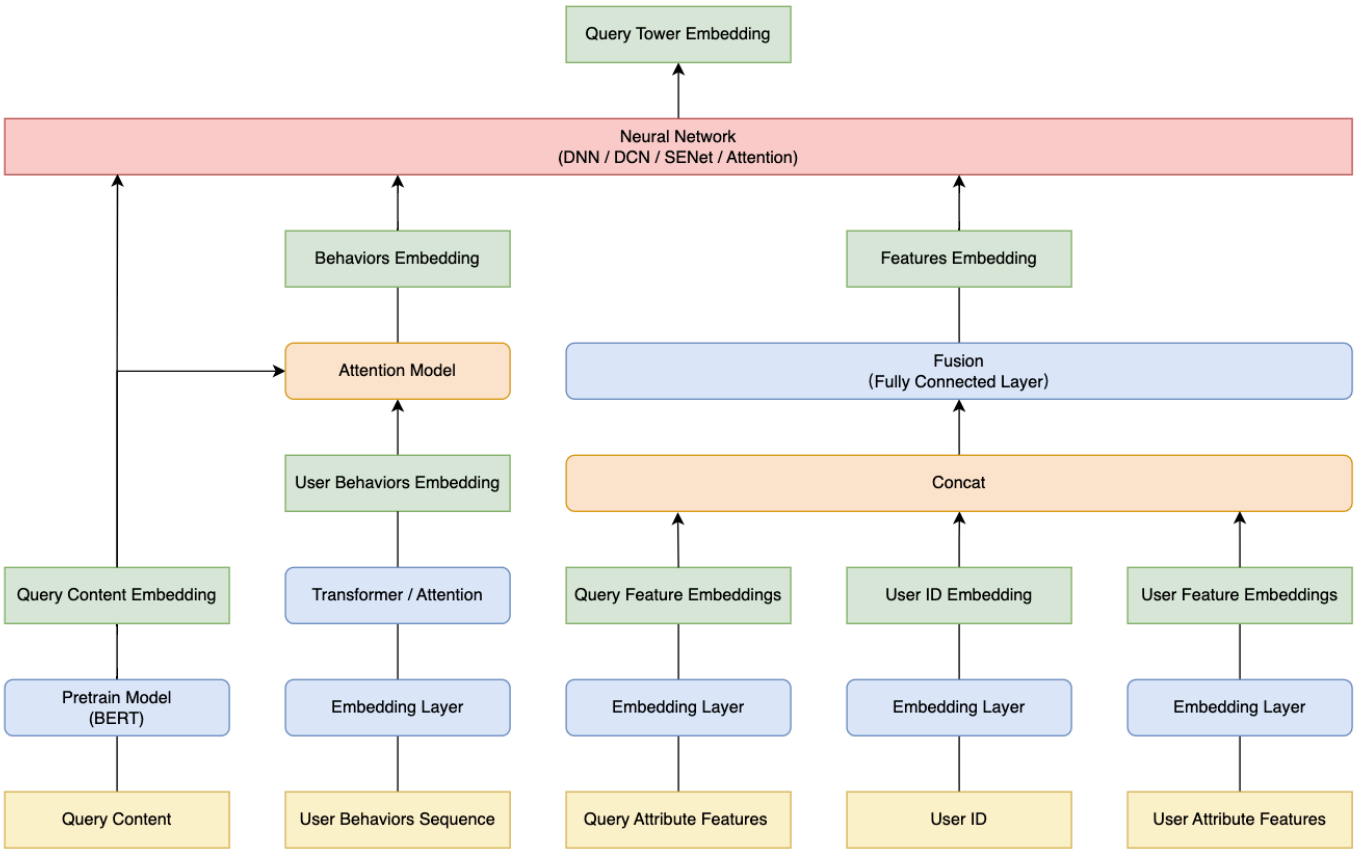
其中，

- Query文本 和 Query属性 中可以用文本表示的特征通常采用 **Sentence-BERT** 转化成向量表示（根据平台算力选择冻结BERT直接采用预训练向量，或随双塔模型同时训练BERT参数）
- 对于用户历史搜索词/点击文档序列，采用 **注意力机制（Attention）** 捕捉用户历史行为和查询词Query的相关性
  - 在序列中添加一个全零向量，模型在进行注意力计算时，可以选择将该零向量赋予最大注意力，可以避免在历史行为与当前查询完全不相关的场景中，模型强制关注到历史行为而带来的噪声
  - 对于不同时间周期的行为序列（实时、短期、长期）可以设计不同的模型策略
- User ID、离散化后的统计特征（分桶）等 **离散特征** 通过Embedding层进行编码，然后可以接入 **全连接层** 将输入特征的各个维度组合起来，并通过权重矩阵进行线性变换，使得模型可以捕捉到更复杂的特征关系

- User ID 特征可以建模关联到用户的历史行为、偏好以及其他相关信息，可以帮助模型做出 **个性化预测**
- User ID 特征具有高纬度和稀疏性，模型可能会学到不具有泛化能力的噪声而导致过拟合：
  - 使用 **哈希函数** 将ID空间映射到一个较小的哈希空间以减少内存使用
  - 使用 **特征降维** 技术（如主成分分析，PCA）对ID向量进行降维
  - 使用 **Dropout** 随机丢弃部分神经元以防止过拟合
- 对于缺少搜索消费行为的 User ID 特征的 **冷启动** 问题
  - **迁移学习：** 采用其他产品域的 User ID 向量
  - **预训练模型生成：** 利用预训练语言模型从用户的文本信息（如用户注册时填写的描述、行为日志、社交网络数据等）生成与用户相关的向量表示
  - **相似群体代替：** 查找相似且高活的 User ID Embedding 取平均

模型结构

根据模型输入的不同维度的特征以及其各自的属性，需要进行针对性的模型结构设计：



具体的：

- Query Text 通常采用 BERT 转化成向量表示（也可采用文本卷积神经网络替换）
- 用户行为序列采用 Transformer 学习更好的向量表示（Transformer 内部的自注意力机制可以对序列内元素的相互关系进行有效的建模来实现更好的表达）
- 采用 注意力机制（Attention）来捕捉用户历史行为和查询词Query的相关性
  - 在序列中添加一个全零向量，模型在进行注意力计算时，可以选择将该零向量赋予最大注意力，可以避免在历史行为与当前查询完全不相关的场景中，模型强制关注到历史行为而带来的噪声
  - 对于不同时间周期的行为序列（实时、短期、长期）可以设计不同的模型策略

- Query/User 离散特征通过 Embedding层 进行编码，通常对编码后的多个离散向量进行拼接，将信息整合到一个向量中

在得到Query特征和User特征向量表示之后，需要经过神经网络实现特征间的交互，常用的网络结构有DCN、CIN、FiBiNet等，此外通常采用SENet对塔底层各通道的特征进行特征增强和过滤。

文档塔

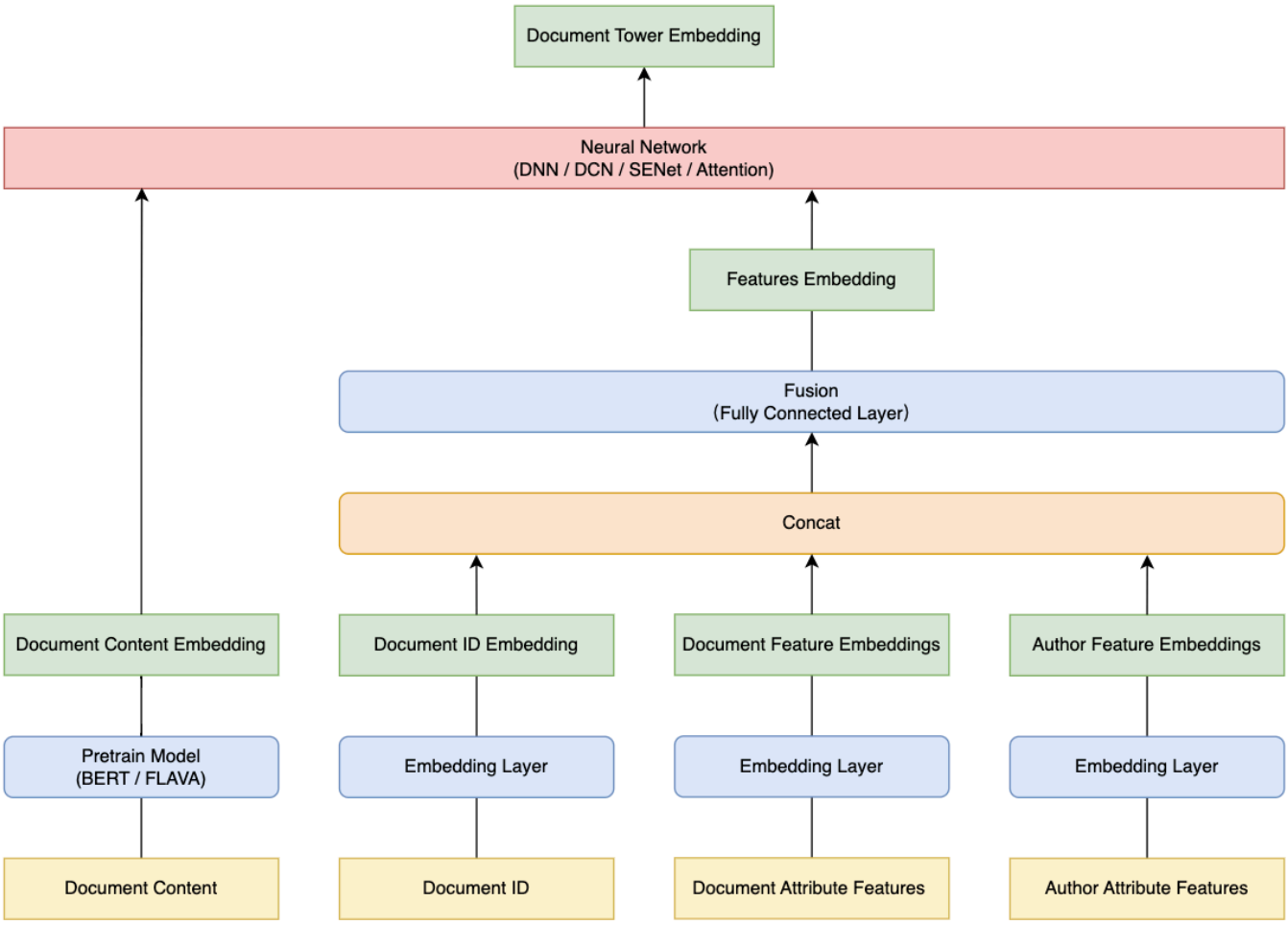
模型输入

查询塔的模型输入特征通常有三个维度：Doc特征（文档特征）、Author特征、Author-Doc交叉特征：

特征	类型
Doc文本（标题/正文/OCR/ASR/评论等）	Document特征
Doc多模态特征（图片/视频/音频等）	
Doc属性特征（Doc ID/类目/标签/主题/点击Query等）	
Doc统计特征（点击/点赞/收藏/评论等统计量）	
Author属性特征（User ID/性别/年龄/所在省市/画像）	Author特征
Author统计特征（发布/曝光/点赞/收藏/评论等统计量）	
Doc-Author交叉特征	Doc-Author交叉特征

模型结构

文档塔的模型结构可以设计为：



特征交叉

传统的双塔中DNN结构一般由多个全连接层组成，每层通过加权和激活函数进行非线性变换，特征交互通过全连接层隐式学习特征交叉。在每一层，输入特征 $\mathbf{x}$ 和权重矩阵 $\mathbf{W}$ 的运算是标量相乘并求和的过程，在捕捉高阶交叉和复杂的特征关系时表达能力有限。如何更好的平衡显式和隐式的特征交叉，以及动态分配特征重要度是模型是提升模型性能的关键因素。下面是常见的特征交叉网络。

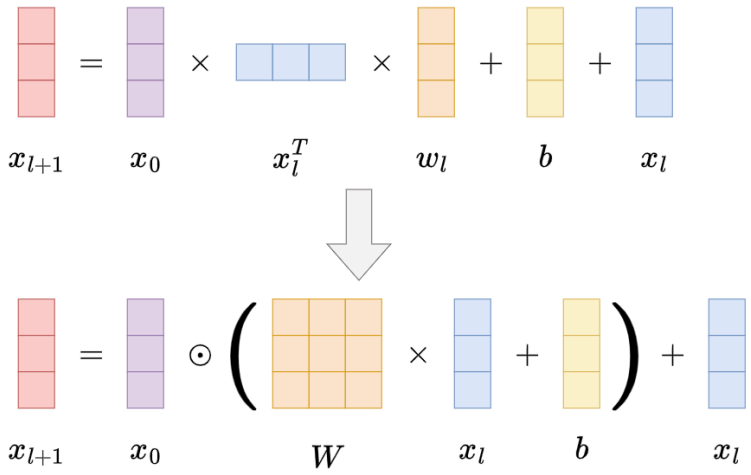
DCN（Deep & Cross Network）

DCN（Deep & Cross Network）结合了 **深度学习（DNN）** 和 **显式的特征交叉（crossing）** 机制，能够有效地捕捉特征之间的高阶交互。DCN通过交叉层显式地学习特征交叉，而传统的DNN需要依靠全连接层隐式学习这些交互。DCN架构分为 Deep Network 和 Cross Network 两部分。

Cross Network

Cross Network 由 **交叉层（Cross Layer）** 构成，交叉层通过直接对输入特征进行交叉（即特征组合）来捕捉特征之间的高阶交互，不需要依赖全连接层逐层学习，从而显式建模输入特征的交互。

Cross Layer 经历了两版变化：



- 第一版：

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^T \mathbf{w}_l + \mathbf{b}_l + \mathbf{x}_l = f(\mathbf{x}_l, \mathbf{w}_l, \mathbf{b}_l) + \mathbf{x}_l \tag{2}$$

  - $x_0 \in \mathbb{R}^d$ 是包含原始特征的最底层输入
  - $x_l, x_{l+1} \in \mathbb{R}^d$ 分别表示第 l 层和第 $l+1$ 层的输出
  - $w_l, b_l \in \mathbb{R}^d$ 表示第 l 层的权重矩阵和偏置参数
  - 每个交叉层在特征交叉 f 后都会将其输入加回，即映射函数 $f: \mathbb{R}^d \mapsto \mathbb{R}^d$ 拟合了 $x_{l+1} - x_l$ 的残差

- 第二版：

$$\mathbf{x}_{l+1} = \mathbf{x_0} \odot (W_l \mathbf{x}_l + \mathbf{b}_l) + \mathbf{x}_l \tag{3}$$

  - $x_0 \in \mathbb{R}^d$ 是包含原始特征的最底层输入
  - $x_l, x_{l+1} \in \mathbb{R}^d$ 分别表示第 l 层和第 $l+1$ 层的输出
  - $W_l \in \mathbb{R}^{d \times d}$ 和 $b_l \in \mathbb{R}^d$ 表示第 l 层的权重矩阵和偏置参数
  - $\odot$ 是哈达玛积运算（Hadamard Product），即逐元素相乘：两个相同大小的矩阵/向量的每个对应元素进行逐一相乘，结果仍是一个维度一致的矩阵/向量，其每个元素是原始矩阵/向量对应元素的乘积

- 第二版本的交叉层采用哈达玛积，向量 $w \in \mathbb{R}^d$ 替换为矩阵 $W \in \mathbb{R}^{d \times d}$，可以实现逐元素的加权，拟合能力加强

- 第一版的交叉层学习的是特殊类型的高阶特征交互，其中每一层都是 x_0 的标量倍数（详细论证见xDeepFM）：
  - $$\begin{aligned}\mathbf{x}_{l+1} &= \mathbf{x}_0\mathbf{x}_l^T\mathbf{w}_{l+1} + \mathbf{x}_l \\ &= \mathbf{x}_0\left(\left(\alpha^l\mathbf{x}_0\right)^T\mathbf{w}_{l+1}\right) + \alpha^l\mathbf{x}_0 \\ &= \alpha^{l+1}\mathbf{x}_0\end{aligned} \tag{4}$$
  - 其中，
    $$\alpha^{l+1} = \alpha^l(\mathbf{x}_0^T\mathbf{w}_{l+1} + 1) \tag{5}$$

    是一个标量，因此 x_{l+1} 仍然是 x_0 的一个标量倍数

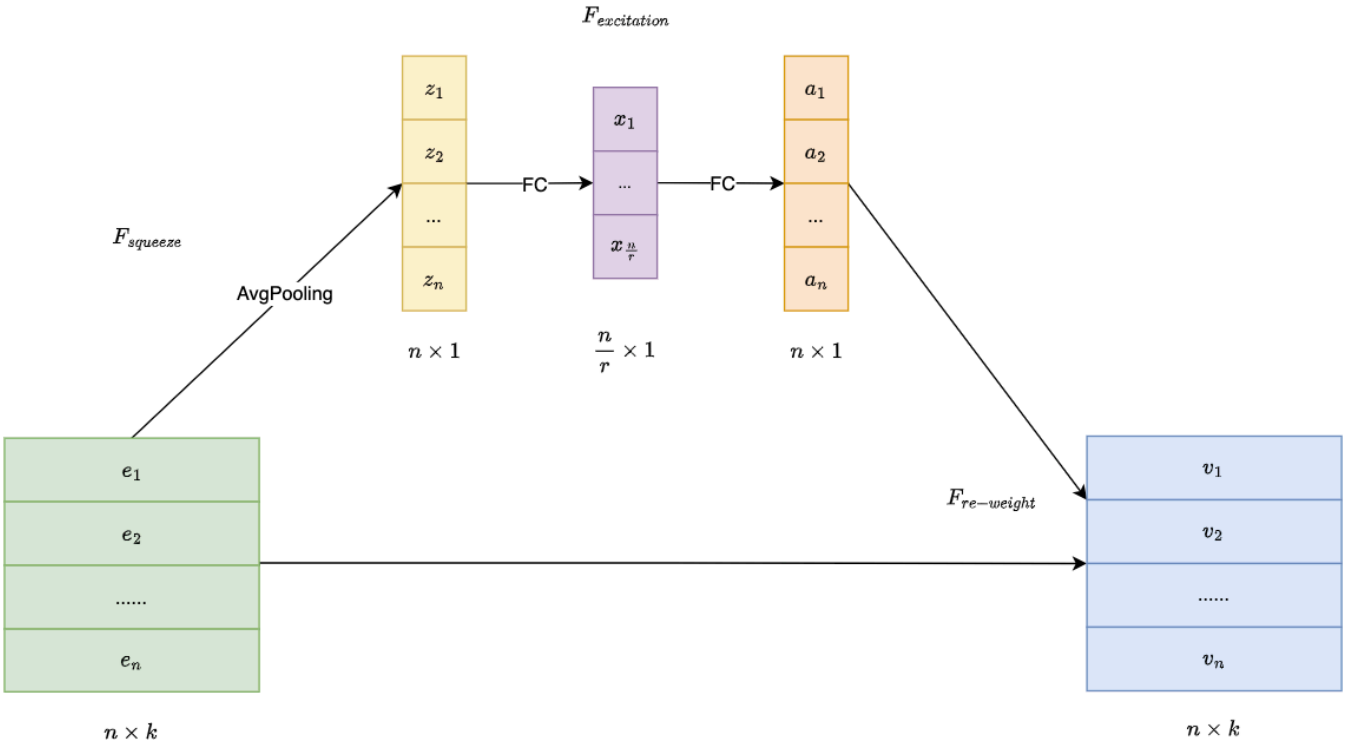
Deep Network

- **深度层（DNN Layer）**：DCN还保留了DNN的设计，即通过多层全连接层（FC层）来学习输入数据的非线性表示。这部分与标准的DNN相似，通过逐层的非线性变换来提取高层次的特征

SENet & FiBiNet

SENet

SE-Net 引入 **Squeeze-and-Excitation (SE) Block** 来增强神经网络表示能力的架构。SE块通过自适应地调整各通道的权重，从而提高网络对有意义特征的关注能力。



- **Squeeze**：通过全局平均池化（Global Average Pooling）对每个通道的特征压缩为一个标量，生成一个通道描述符。即对每个通道进行信息整合，得到通道的全局特征表示
  - $$z_i = F_{sq}(e_i) = \frac{1}{k}\sum_{t=1}^{k} e_i^{(t)} \tag{6}$$
  - 其中有 n 个特征 e，每个特征可表示为k个向量（域，filed）

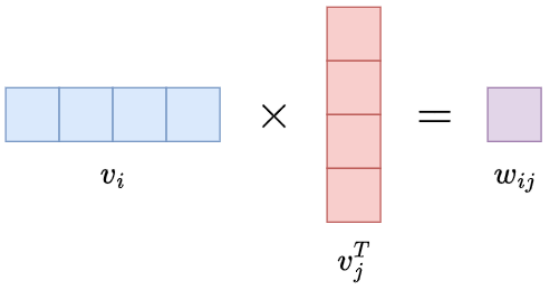
- 该步骤将原始向量 $E=[e_1,\ldots,e_n]$ 挤压成统计向量 $Z=[z_1,\ldots,z_n]$，z_i 为标量

- **Excitation**：通道描述符通过一个小型的全连接网络（通常是一个两层的 MLP）进行进一步处理，生成每个通道的权重（或缩放因子）。即为每个通道分配一个重要性权重，增强重要通道，抑制不重要的通道
  - $$A=F_{ex}(Z)=\sigma_2(W_2\sigma_1(W_1Z)) \tag{7}$$
  - Excitation采用两个全连接层（FC）学习权重，第一个FC层是降维层，参数为 W_1，降维的压缩比是超参数 r；第二个FC层通过参数 W_2 恢复维度
  - σ_1 和 σ_2 为非线性激活函数，通常分别用 ReLU 和 Sigmoid
  - 第一层通过压缩层减少维度，从而限制模型复杂度并帮助泛化，同时也使得模型聚焦于更有意义的通道依赖。采用 ReLU 引入非线性，学习通道之间的相互依赖性
  - 第二层则将这种压缩后的信息恢复，并学习到每个通道的重要性。通过 Sigmoid 函数将输出映射到 [0,1] 的范围，为每个通道分配一个注意力系数。

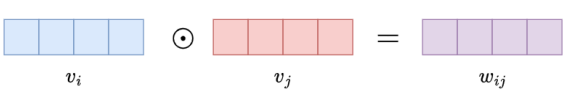
- **Re-Weight**：对原始域向量 E 和域权重向量 A 进行 **逐域（field-wise）相乘**，输出新的向量 $V=[v_1,\ldots,v_n]$，**域（field）** 表示一个特征的向量
  - $$V=F_{ReWeight}(A,E)=[a_1\cdot e_1,\cdots,a_n\cdot e_n]=[v_1,\cdots,v_n] \tag{8}$$

双线性交互层（Bilinear-Interaction Layer）

SENet实现了特征（field）加权，而交互层则实现了特征交叉。交互层是一个计算 **二阶特征交互** 的层（二阶交互描述了两个特征之间的相互作用，而非独立贡献），特征交互的方法有：

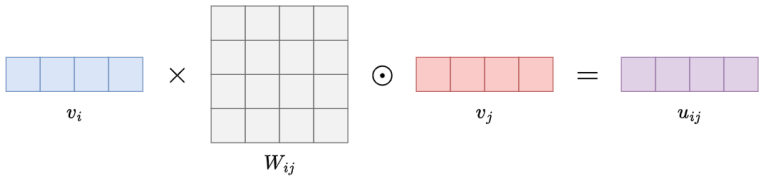
- **内积（Inner Product）**
  - 两个向量相乘得到一个标量（实数）的运算，内积的几何意义是两个向量的乘积与它们夹角的关系
  - 

  - $$\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_i \tag{9}$$

- **哈达玛积（Hadamard Product）**
  - 两个相同大小的矩阵（或向量）的每个对应元素进行逐一相乘，结果仍然是一个矩阵（或向量），其每个元素是原始矩阵（或向量）对应元素的乘积
  - 

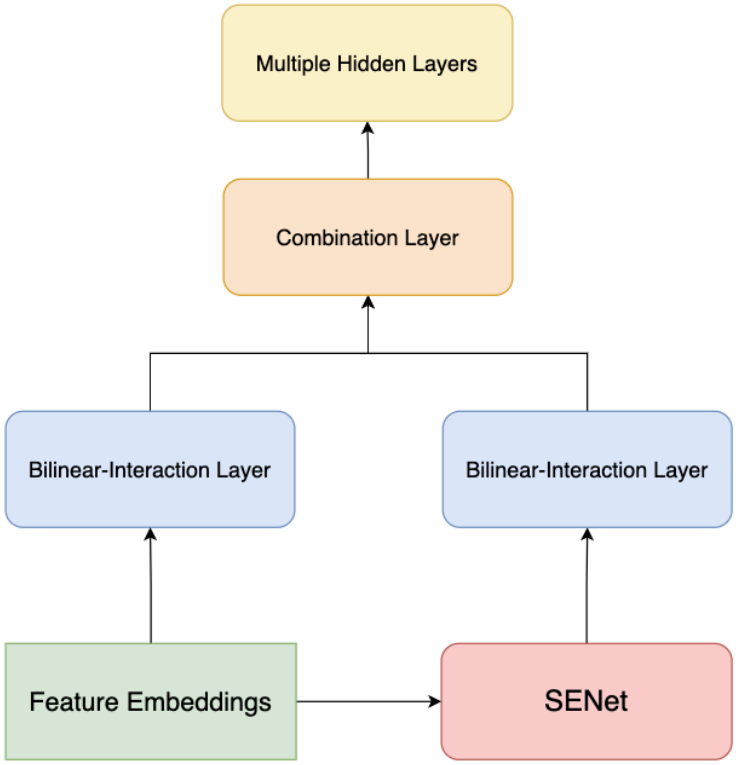
  - $$\mathbf{a}\circ\mathbf{b}=[a_1b_1,a_2b_2,\cdots,a_nb_n] \tag{10}$$

- **双线性交互（Bilinear Interaction）**
  - 通过计算两个特征向量之间的双线性变换，捕捉它们的交互信息：$\text{Bilinear fnteraction} = \mathbf{x}^{\mathrm{T}}\,\mathbf{W}\mathbf{y}$
  - 其中 $\mathbf{W}$ 是权重矩阵，表示两个特征空间之间的交互关系
    - 若 $\mathbf{W}$ 是 **对角矩阵**，则 bilinear interaction 退化为元素级交互
    - 若 $\mathbf{W}$ 是 **稀疏矩阵**，可以高效地捕获重要的交互关系
    - 若 $\mathbf{W}$ 是 **全矩阵**，则可以完全表达两个向量的任意关系，但代价是更高的计算复杂度
  - 在FiBiNet中，双线性交互的实现方式是结合了内积和哈达玛积，如图：
  -



FiBiNet

FiBiNet 的模型结果如下：



原始特征向量经过 Bilinear-Interaction 得到向量 p，另一路中原始向量经过 SENet 后在通过 Bilinear-Interaction 得到向量 q，将两个向量拼接（Concat）后得到向量 c 并给到下游 DNN 进行预测。

训练样本构建

众所周知，数据和特征决定了算法的上限，模型和算法只是逼近这个上限。训练样本构建直接影响模型的效果，构建训练集需要平衡正负样本的质量和多样性。合理的采样策略和增强技术、负样本采样的优化和不平衡问题的处理都是训练样本构建的关键因素。

样本选择偏差

离线训练的数据分布直接决定了模型的泛化能力和对真实线上场景的适配性，**训练集的数据分布需要与在线服务时的数据分布保持一致**。

基于用户搜索曝光点击数据构建正负样本是常见思路，但是在召回阶段过度采用曝光/点击数据则会带来样本选择偏差问题。具体的，不同于粗排/精排阶段，召回阶段面对全局Doc库，而曝光/点击采样的Doc是经过后链路筛选（粗排/精排）和用户行为驱动的，并非全局Doc语料库。仅仅采用曝光/点击样本会导致召回模型的训练结果对部分数据的过拟合，进而影响模型的泛化能力，在面对全局物料库时性能下降。

样本构建方法

训练样本的数据格式为 Query-Doc 对，常见的样本挖掘方法如下：

- **曝光点击-正样本**
  - 搜索日志中检索Query后的点击Doc随机采样构建 Query-Doc 作为正样本
- **粗排/精排过滤-负样本**
  - 进入粗排或精排阶段，因打分低被截断过滤的Doc（未曝光）与检索Query构建 Query-Doc 负样本
- **全库随机采样-负样本**
  - 在原始的全局物料库里，按**曝光频率**随机抽取Doc和检索Query构建 Query-Doc 负样本
- **Batch内随机采样-负样本**
  - 在只包含正样本的一个训练Batch内，当前正样本 Query-Doc 中的Query与其他样本的Doc构建 Query-Doc 负样本
- **困难样本**
  - 困难样本能带来更多的信息量，对难以区分的样本的学习会让模型更好地定义**决策边界**
  - **难正样本**
    - 采用语义相同的Query（同义替换/丢弃非核心词）替换原检索词以构建 Query-Doc 正样本
  - **难负样本**
    - 去除Query中的核心词（相关性信号重要的词）以构建 Query-Doc 负样本
    - **插值生成：**
      1. 从全库中随机采样一批负样本，与当前用户查询Query向量计算相似度
      2. 通过相似度分数筛选出与Query向量最相似的Top N负样本
      3. 在Top N负样本向量和正样本向量之间进行**线性插值**。假设正样本向量为$\mathbf{v}_+$，负样本向量为$\mathbf{v}_-$，则插值生成的难负样本向量表示为$\mathbf{v}_{\mathbf{hard}}$：

$$\mathbf{v}_{\mathbf{hard}} = \alpha \mathbf{v}_+ + (1-\alpha)\mathbf{v}_- \tag{11}$$

      其中，$\alpha \in [0,1]$是控制插值比例的超参数，通常从均匀分布中随机采样。可通过动态调整α的分布来控制训练阶段的难度，例如，训练初期可以采用较小的α值，逐渐增大难负样本的比例。

在线上真实索引环境中，召回是从海量不相关Doc中找到相关结果，所以模型面对的更多是容易区分的数据。因此，负样本主要靠随机采样生成简单负样本，难负样本作为补充，比例通常维持在 Easy : Hard = 100 : 1。

优化目标定义

召回常用的损失函数是多分类的 **Softmax Loss**，相比于侧重局部比较、优化相对关系的 **Hinge Loss** （如：Pairlist），交叉熵损失在训练和推理阶段可保持一致性，并使得模型具备 **全局比较能力**。

NCE Loss

Softmax Loss 把召回目标看成一个多分类问题：

已知交叉熵函数为：

$$-\sum_{i=1}^{k} y_i * log(\hat{y}_i) \tag{12}$$

当采用 Softmax 作为激活函数时，通常由于标签 y_i 是 one-hot 编码，只有目标正确标签 $y_i = 1$，其余为 $y_i = 0$，则交叉熵损失函数可简化为只关注 $y = 1$ 的部分。

给定查询向量 q 和一组文档向量 $\{d_1, d_2, \ldots, d_N\}$，Softmax Loss 可表示为：

$$\mathcal{L}_{\text{softmax}} = -\log \frac{\exp(q \cdot d^+)}{\sum_{i=1}^{N} \exp(q \cdot d_i)} \tag{13}$$

其中，q 是查询向量，d^+ 是与查询相关的正样本，d_i 是候选文档向量，N 是检索库中所有文档的数量，学习目标是从候选文档中选中 d^+ 的概率最高。

由于每篇Doc被看成一个类别，而因此损失函数的分母要计算Query向量和所有Doc向量的点积，为了优化这种超大规模分类问题，通常采用 **NCE Loss（Noise Contrastive Estimation）** 这种通过噪声对比学习的方式来 **近似估计Softmax** 的方法。

NCE Loss 通过引入 **噪声样本**（负样本）来简化损失函数的计算，进而 **近似计算** 正样本与负样本之间的相似度，而不需要对所有文档进行归一化。其核心思想是通过对比数据分布中的真实样本（正样本）与噪声分布中的样本（负样本）来估计数据分布的参数。

NCE 的损失函数如下：

$$\mathcal{L}_{NCE} = -\sum_{(q,d) \in B} (\log \sigma(q \cdot d^+) - \sum_{i=1}^{K} \log \sigma(-q \cdot d^-)) \tag{14}$$

其中，d^+ 是正样本，d^- 是负样本，K 是负样本数量，σ 是 sigmoid 激活函数，B 代表一个Batch。

与Softmax相比，NCE的计算量大大减少，因为它只需要计算正样本和一小部分负样本的相似度，而Softmax的计算复杂度是与文档库大小 N 成正比的。NCE基本思想是通过与噪声样本（负样本）进行对比，来近似估计正样本的概率。

InfoNCE Loss

NCE 将问题看作 **二分类任务**，其中正样本属于一个类，而负样本属于噪声类。但是，负样本是随机选择的无关文档，可能彼此属于完全不同的类别。因此，负样本的多样性和分布可能导致模型的学习过程不稳定或者收敛速度变慢。

InfoNCE 是 NCE 的一个变种，引入了 **多分类问题**的思想，特别是通过 **负样本数量** 来设定多分类的目标。因此，InfoNCE 将 **二分类问题** 转变为 **多分类问题**，其中每个负样本相当于一个独立的类别。

InfoNCE 目标是 **最大化正样本与查询样本之间的相似度**，同时 **最小化查询样本与多个负样本之间的相似度**。给定查询样本 q，负样本 d^-，正样本 d^+，InfoNCE Loss 可表示为：

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(q, d^+)/\tau)}{\exp(\text{sim}(q, d^+)/\tau) + \sum_{j=1}^{K} \exp(\text{sim}(q, d_j^-)/\tau)} = -\log \frac{\exp(\text{sim}(q, d^+)/\tau)}{\sum_{i=0}^{N} \exp(\text{sim}(q, d_i)/\tau)} \tag{15}$$

其中，sim 是相似度度量函数（COSINE/点击），τ 是温度参数，用于控制模型对不同相似度分数的敏感度，当 τ 设置的很小，则模型会关注困难负样本，泛化能力变差，反之损失会对所有负样本区分能力变弱。

在忽略温度参数 τ 下，InfoNCE 就是交叉熵损失（CE，Cross Entropy Loss），唯一区别在 CE 中 N 为数据集里所有类别数量，而 InfoNCE 中 N 为 K 个负样本的数量加 1 个正样本数量。而引入随机负采样的 **Sampled Softmax Loss** 和 InfoNCE 是基本一致的：

$$\mathcal{L}_{\text{Sampled Softmax}} = -\log \frac{\exp(\text{sim}(q, d)/\tau)}{\sum_{i=1}^{N} \exp(\text{sim}(q, d_i)/\tau)} \tag{16}$$

Triplet Loss

Triplet Loss 属于 **Hinge Loss**，其目标是使得锚点（Anchor）与正样本（Positive）之间的距离尽量小，同时使锚点与负样本（Negative）之间的距离尽量大。

Triplet Loss 函数如下：

$$\mathcal{L}_{\text{Triplet}} = \max(\text{sim}(q, d^+) - \text{sim}(q, d^-) + \alpha, 0) \tag{17}$$

其中，q 是锚点查询样本 Query，d^- 为负样本，d^+ 为正样本，sim是距离度量函数（一般采用1-COSINE），α是超参数，表示距离相差边际。

> $\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$，$A$和$B$是两个向量，$\|A\|$和$\|B\|$是它们的L2范数，余弦相似度的值范围为$[-1, 1]$，余弦相似度越大，表示两个向量越相似（余弦相似度衡量的是两个向量之间的夹角，反映的是它们的方向相似度）

多目标与目标一致性

召回作为粗排和精排的前置环节，其任务是为粗排、精排提供适合的候选项。通常在召回模型的训练中，点击行为的样本通常被视为正样本，而点击作为正样本的做法在某些情况下可能存在一定的局限性。

通过将深度消费指标（如点赞、收藏、评论等）引入召回模型，模型能够为精排提供具有更高深度消费潜力的候选项，确保了召回和精排阶段的目标不会偏离，前置环节的候选集更能满足精排的需求，从而避免精排时的搜索效率损失。

设模型输出为 $\hat{y}_i$，表示对某个候选项 i 的预测评分。目标函数 L 可以写成多任务学习的形式：

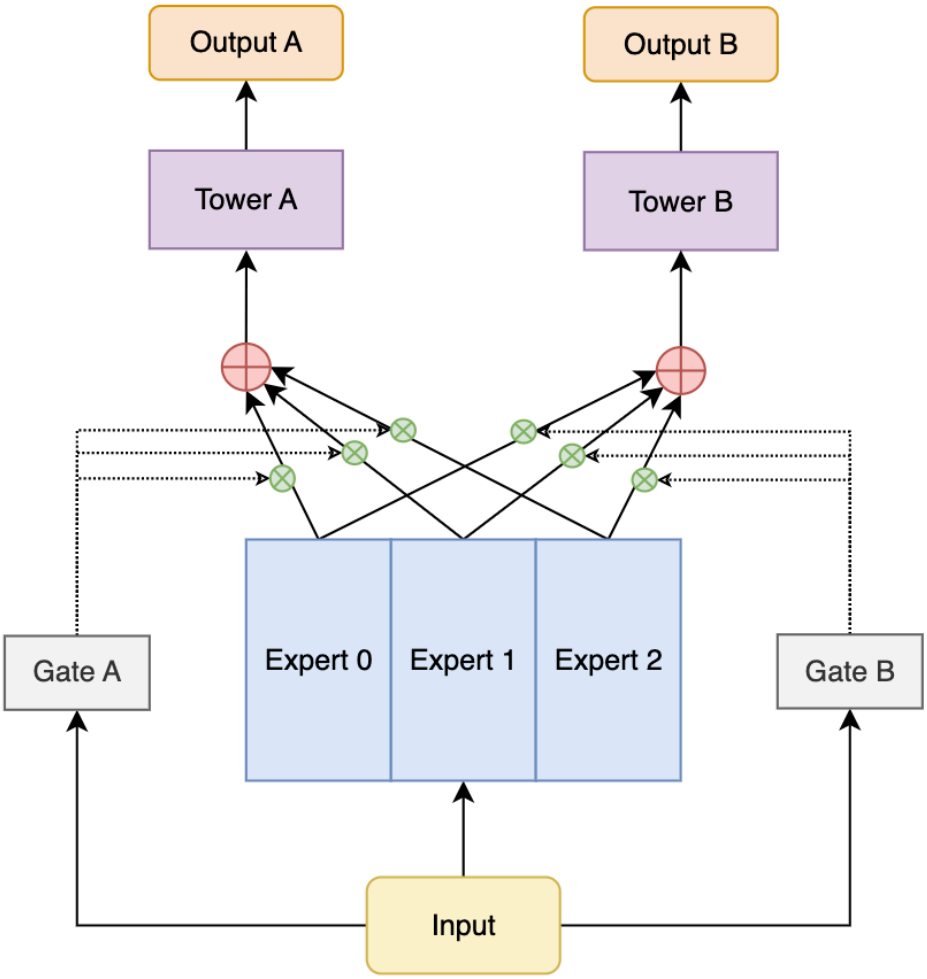
$$L = \lambda_c L_c(\hat{y}_i, Y_c) + \lambda_l L_l(\hat{y}_i, Y_l) + \lambda_s L_s(\hat{y}_i, Y_s) + \lambda_r L_r(\hat{y}_i, Y_r) \tag{18}$$

其中：

- L_c，L_l, L_s, L_r 分别表示点击、点赞、收藏、评论的损失函数
- λ_c，λ_l，λ_s，λ_r 是各个目标的权重参数，用于平衡不同目标的贡献

MMoE

常见的多目标预估模型有MMoE（Multi-gate Mixture of Experts），任务 T_k 的最终输出由多个共享专家网络 $(e_1, e_2, \ldots, e_N)$ 的加权组合生成，权重由任务特定的门控网络 g_k 动态计算。



专家网络

每个专家网络的输入为通用特征 x，经过专家网络 e_i 后得到输出 h_i：

$$h_i = e_i(x), \quad i = 1, 2, \ldots, N \tag{19}$$

其中：

- e_i 是第 i 个专家网络
- $h_i \in \mathbb{R}^d$ 是第 i 个专家网络的输出

门控网络

任务 T_k 的门控网络根据输入特征 x，计算每个专家的权重 $g_{k,i}$：

$$g_{k,i} = \text{softmax}(W_k \cdot x + b_k)_i \tag{20}$$

其中：

- $g_{k,i} \in [0, 1]$ 是专家 e_i 对任务 T_k 的权重
- W_k 和 b_k 是门控网络的参数

专家输出融合

任务 T_k 的最终共享特征输出 o_k 是所有专家输出的加权和：

$$o_k = \sum_{i=1}^{N} g_{k,i} \cdot h_i \tag{21}$$

其中：

- $g_{k,i}$ 是任务 T_k 对专家 e_i 的权重
- h_i 是专家 e_i 的输出

任务预测

共享特征 o_k 传递到任务特定的预测网络 f_k，得到最终的任务预测结果：

$$\hat{y}_k = f_k(o_k) \tag{22}$$

其中 f_k 是任务 T_k 的特定网络。

线上推理和模型更新

ANN

向量召回的检索过程是将查询经过向量化处理，利用查询向量与所有文档向量之间的相似度进行计算实现检索，即查找与查询向量最接近的TopN个文档向量，文档向量对应的文档作为召回结果。

由于在大规模数据集上计算全量的相似度非常耗时，为了避免在大规模文档库中进行 **暴力枚举**（即对每个文档进行逐一比较），通常采用 **近似最近邻搜索（ANN，Approximate Nearest Neighbor Search）** 技术来加速线上召回推理过程。使用 ANN 索引结构，可以大大提高向量检索的效率，在不需要对所有文档进行完整的搜索的情况下，快速找到最相似的文档。

1. **文档向量化**：通过双塔模型的 **文档塔**，将所有文档（或物料）映射为向量
2. **构建向量索引**：将这些文档向量存储到一个高效的索引结构中，常见的向量索引方法包括：
   - **HNSW（Hierarchical Navigable Small World）**：一个基于图结构的索引方法，特别适用于高维向量的近似最近邻搜索
   - **FAISS（Facebook AI Similarity Search）**：一个高效的向量检索库，支持多种索引方法，如 **IVF（Inverted File）**、**PQ（Product Quantization）**、**HNSW** 等
   - **Annoy**：用于构建树状结构的向量索引，支持高效的最近邻搜索
3. **查询向量化**：通过 **查询塔**，将用户输入的查询Query转换为一个嵌入向量
4. **搜索最近邻**：使用 ANN 索引（如 FAISS、HNSW）从 **文档库的向量索引** 中检索出与查询向量最相似的TopN个文档（即与查询向量最近的向量）

IVFFLAT & IVFPQ

IVFFLAT 和 IVFPQ 是常见的ANN索引方法，通过将向量分组和压缩，实现高效地检索大规模高维向量数据。

IVFFLAT

基本原理：

1. **聚类分组**：通过聚类算法（例如 k-means），将数据库中的向量划分成 k 个簇，每个簇对应一个质心（centroid）。这些质心被存储在一个倒排文件（Inverted File）中
2. **倒排索引**：每个向量根据与质心的距离被分配到某个簇（倒排列表）
3. **精确检索**：
   - 查询时，计算查询向量到所有质心的距离，选取与查询向量最近的几个簇
   - 仅在这些簇中的向量上计算精确的距离（例如欧氏距离或余弦相似度）
4. **返回结果**：按距离排序并返回最近邻结果。

优点：

- **高效检索**：通过分簇减少需要比较的向量数量
- **精度高**：保留了向量的原始信息（无压缩）

缺点：

- 对内存需求较高：每个向量以原始形式存储在簇中
- 在超大规模向量库中，存储和查询效率可能会下降

IVFPQ

基本原理：
IVFPQ 是在 IVFFLAT 的基础上，利用 **Product Quantization（PQ）** 对簇中的向量进行压缩存储，从而进一步降低内存需求和检索时间。

1. **聚类分组**：
   - 和 IVFFLAT 类似，先用 k-means 聚类将向量分配到 k 个簇
2. **量化存储**：
   - PQ 将每个向量分割成多个子向量，并对每个子向量独立地进行量化（例如使用多个子码本）
   - 每个子向量被编码为一个较小的编码值（比如用 8 位整数表示）
3. **倒排索引**：
   - 索引记录的不是原始向量，而是压缩后的 PQ 编码
4. **查询过程**：
   - 先找到查询向量最近的簇。
   - 在选中的簇中，对所有 PQ 编码的向量进行快速的距离近似计算，找到最接近的向量

优点：

- **高效存储**：向量被压缩为低维的编码，极大降低存储需求
- **快速检索**：通过量化加速距离计算
- **灵活性**：可以调节簇数和 PQ 的量化程度以平衡速度和精度

缺点：

- 量化带来一定的精度损失，尤其是在向量库中向量分布不均匀时
- 构建索引和更新索引的成本较高

模型更新

随着查询词Query和文档Doc之间的交互不断发生（如点击、购买等行为），模型需要定期更新以保持对用户偏好和文档特征的实时适应。天级别的增量更新方法是一种常见的策略，旨在通过不断学习新的用户行为数据，逐步调整模型的参数，保持模型的有效性。

1. **数据收集和预处理**：每天收集新的用户搜索行为数据并构建正负训练样本
2. **增量训练（Incremental Training）**：基于当前天的新增数据，对模型进行增量训练。增量训练在现有模型基础上只进行小范围的调整，而不是从头开始训练。从而避免重新计算全部数据和参数，提高训练效率
3. **模型参数更新**
   - **在线学习（Online Learning）**：通过实时数据流实时更新模型参数，而不是批量更新，来实现对新数据的实时适应

    - **离线学习（Offline Learning）**：当数据量足够大时，可以定期（日/周/月）对模型进行全面的训练。定期的批量训练（如每月）确保模型对整个数据分布的广泛适应，避免过拟合最近的数据，而忽略了历史趋势
4. **离线推理和重建索引**：基于更新参数后的模型离线对检索文档计算得到向量并以此重建索引
5. **模型部署与在线推理**：将更新后的模型部署到线上环境中，在用户查询时，使用更新后的模型进行实时的在线推理获得查询向量

突破双塔

双塔模型通过分离查询 Query 和文档 Doc 信息的处理，允许预先计算并索引文档向量，并通过 ANN 实现高效快速的召回。但是双塔模型的查询塔和文档塔相互独立，无法直接利用 Query-Doc 之间的交互信息造成信息交互不足。另外，仅通过查询向量与文档向量的内积来衡量两者之间的相关性，模型难以精确地表达两者之间更细微或非线性的关联，造成表达能力受限。

目前业内针对双塔模型，在海量数据召回保持低耗时的前提下，通过各种改进方案提升模型建模能力：

- **引入索引结构**：比如树形索引或者图结构索引，这些结构可以帮助提高搜索效率，同时使得能够更加灵活地表达用户-物品间的关系。
- **联合优化模型与索引**：通过设计可以考虑索引结构约束的模型，不仅可以保证高效的检索速度，而且能让模型更好地学习到满足特定索引要求的数据分布
- **对偶增强双塔**：在各自的塔输入特征中引入了一段增广向量，用来表征另一个塔的信息，从而通过隐层计算实现一定程度上的特征交叉

总结

向量召回是一种基于向量空间模型进行信息检索的方法，核心思想是将查询Query和文档Doc转换为向量，通过计算这些向量之间的相似度来进行检索。向量召回通常采用双塔模型，为了实现特征交叉，一般采用 DCN、Attention等网络架构。在训练样本构建上，采用随机负采样生成大量负样本，并通过 Softmax Loss 、Hinge Loss 作为目标损失函数。

向量召回是召回体系的重要组成部分，且其技术上限非常高，需要有明确的指标作为策略引导，建设理想测试集是评估和优化策略的关键。算法模型的训练样本构建、特征挖掘和模型设计需要充分考虑真实在线的数据分布，在做好召回的基本定位的同时充分利用用户行为数据，尽可能满足搜索整体业务目标以保持链路一致性。此外，需要在模型复杂度、策略时效性、迭代成本、计算开销、在线推理延迟中不断权衡，在平衡召回效果和系统效率之间根据当前业务需求做出灵活调整。

参考文献

1. Embedding-based Retrieval in Facebook Search
2. Embedding-based Product Retrieval in Taobao Search
3. Multi-Objective Personalized Product Retrieval in Taobao Search
4. MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search
5. FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction
6. Dense Text Retrieval based on Pretrained Language Models: A Survey
7. Deep Interest Network for Click-Through Rate Prediction
8. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems
9. Deep & Cross Network for Ad Click Predictions
10. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems
11. Squeeze-and-Excitation Networks
12. Notes on Noise Contrastive Estimation and Negative Sampling
13. Learning Tree-based Deep Model for Recommender Systems

14. Approximate Nearest Neighbor Search under Neural Similarity Metric for Large-Scale Recommendation
15. Beyond Two-Tower Matching: Learning Sparse Retrievable Cross-Interactions for Recommendation
16. Deep Retrieval: An End-to-End Learnable Structure Model for Large-Scale Recommendations
17. I^3 Retriever: Incorporating Implicit Interaction in Pre-trained Language Models for Passage Retrieva
18. A Dual Augmented Two-tower Model for Online Large-scale Recommendation
19. Multivariate Representation Learning for Information Retrieval
20. Beyond Two-Tower: Attribute Guided Representation Learning for Candidate Retrieval
21. Multi-Aspect Dense Retrieval
22. VIRT: Improving Representation-based Text Matching via Virtual Interaction
23. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts
24. Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations