召回是负样本的艺术

排序、召回，我感觉待遇还是相差蛮大的：

- 排序
- - 排序，特别是精排，处于整个链条的最后一环，方便**直接对业务指标发力**。排序一般都是多目标预估，预测item的CTR、CVR、观看时长等指标。比如我们想要提升观看时长的指标，就在排序的打分公式中赋予“观看时长”以较高的权重。
  - 排序由于候选集较小，所以有时间使用复杂模型，各种NN，各种Attention，能加的，都给它加上，逼格顿时高大上了许多。
  - 用于排序的基础设施更加完善，特征抽取、指标监控（直接可以看到CTR/CVR的变化）、线上serving都相对成熟。
- 召回
- - 一个系统中不可能只有一路召回，多路召回之间互相重合，导致单路召回对大盘的影响有限；好处是，相互冗余，有几路出问题，也不至于让排序饿肚子。现在主流比较偏好**向量化召回**。至于怎么生成向量有很多种方法，比如FM/双塔等，本质都是线上做近邻索引。
  - 召回处于整体推荐链条的前端，其结果经过粗排、精排两次筛选，作用于最终业务指标时，影响力就大大减弱了。
  - 受限于巨大的候选集和实时性要求，召回模型的复杂度受限，不能上太复杂的模型。平日里，最简单的基于统计的策略，比如**基于item的协同过滤、基于user的协同过滤**表现就很不错，改进的动力也不那么急迫。

然而，"召回不存，排序焉附"。2020年Facebook最新的论文《Embedding-based Retrieval in Facebook Search》（EBR）非常全面，涵盖了一个召回从**样本筛选**、特征工程、模型设计、**离线评估**、**在线Serving**的全流程。

召回基本流程

- 模型
- - 老生常谈的双塔模型。双塔避免底层就出现特征交叉，方便提前将item embedding存入FAISS。
  - 只要<user,doc>的匹配得分为user embedding/item embedding的**点积**或**cosine**的形式。其实，cosine就是把user/item embedding做归一化之后的点积嘛！
- LOSS
- - **Pairwise Hinge Loss**. loss=max(0,margin-<u,d+>+<u,d->)。即同一个用户与“正item”（点击过的item）的匹配度<u,d+>，要比用户与“负item”的匹配度<u,d->高于一定的阈值。这个margin怎么去选也是一个关键的超参数！
  - **Sampled Softmax Loss.** Youtube/DSSM模型都是使用的Sampled Softmax Loss. 对于每个<user,doc>对，都经过Negative Sampling取得若干个负样本d-，然后在正样本和负样本的得分上做softmax。我们希望分子（<u,d+>得分）尽可能大，而分母（所有的<u,d->得分）尽可能小。
  - **BPR loss**。 但是，根据我的实践经验，使用BPR loss=log(1+exp(<u,d->-<u,d+>))，效果要比这里的Pairwise Hinge Loss好
  - - 文中也说了margin对于模型效果影响巨大，BPR loss**少了一个需要调整的超参**。

        - Hinge loss中，<u,d+>-<u,d->大于margin后，对于loss的贡献减少为0。而BPR则没有针对<u,d+>-<u,d->设定上限，**鼓励优势越大越好。**
-
    - 尽管不同召回算法有不同的loss，但是背后基于Pairwise的思想都是共通，这一点与排序时采用binary cross entropy loss有较大的不同。
- 【召回的离线评估】
-
    - 由于线上AB test周期长、成本高，因此，在线上AB测试之前，我们需要进行充分的离线评估，减少实验的时间成本。
    - 文中所使用的方法是，拿**Top K召回结果**与**用户实际点击**做交集，然后计算precision/recall。这个其实是没有考虑位置信息的，类似hit rate。
    - Airbnb所使用的方法是看“用户实际点击”在“召回结果”中的平均位置，就是**将位置信息考虑进去**，类似于MRR。
    - 实际上不同于排序，召回的离线评测更加难做，因为召回的结果未被用户点击，未必说明用户不喜欢，而可能是由于**该召回结果压根没有给用户曝光过**！
    - 根据我的亲身实践，无论是precision/recall还是平均位置，**和实际线上结果之间还是存在一定gap。**曾经发生过，新模型的离线评测结果没有显著提升，但是上线后去发现提升明显。所以，**缺乏置信度高的离线评测手段仍然是召回工作中的痛点。**
    -

重中之重是"筛选（负）样本"

如果说排序是特征的艺术，那么召回就是样本的艺术，特别是负样本的艺术。样本选择错了，那么上述的模型设计、特征工程，只能是南辕北辙，做得越卖力，错得越离谱。本文在“样本筛选”上的贡献有三：

为什么不能（只）拿"曝光未点击"做负样本

排序是非常讲究所谓的“**真**负”样本的，即必须拿“曝光未点击”的样本做负样本。为此，还有所谓above click的作法，**即只拿点击item以上的未点击item做负样本。**这是因为可能曝光的物品用户根本没有看到（排在后面，这也是一种position bias），那么用户未点击不能代表他不喜欢。反而，在点击item上面的item用户大概率已经看到了，但是还是未点击，说明是真的不喜欢。

如果直接拿“曝光未点击”样本做负样本训练召回模型（就相当于把排序模型改成双塔而已），离线AUC达到0.7+，做为一个特征简化的召回模型，已经算是非常高了，但是线上表现一塌糊涂。离线评测时，发现召回的物料与用户画像、用户点击历史，完全没有相关性。而当我抱着“照着论文画瓢”拿随机采样的样本做负样本，线上结果却非常好。

何也？？？其实，是因为我自以为是的做法，违反了机器学习的一条基本原则，就是**离线训练数据的分布，应该与线上实际应用的数据，保持一致。**

以前，我们谈到召回与排序的不同，往往只强调速度，即因为召回的候选集更大，所以要求速度更快，所以模型可以简单一些。这是只知其一。另一个方面，起码之前是我没有深刻意识到的，就是

- 排序其目标是“从用户可能喜欢的当中挑选出用户最喜欢的”，是为了优中选优。与召回相比，排序面对的数据环境，简直就是**温室里的花朵。**
- 召回是“是将用户可能喜欢的，和海量对用户根本不靠谱的，分隔开”，所以召回在线上所面对的数据环境，就是**鱼龙混杂、良莠不齐。**

所以，要求喂入召回模型的样本，既要让模型见过最匹配的<user,doc>对，也要让模型见过最不靠谱的<user,doc>对，才能让模型达到"开眼界、见世面"的目的，从而在“大是大非”上不犯错误。

- 最匹配的<user,doc>，没有异议，就是用户点击的样本

- 最不靠谱的<user,doc>，是“曝光未点击”样本吗？显然不是。这里牵扯到一个推荐系统里常见的bias，就是我们从线上日志获得的训练样本，已经是**上一版本的**召回、粗排、精排替用户筛选过的（注：我们在已有的召回模型上继续改进，所以一定要有一个“上一版本的”召回模型），即已经是对用户“比较靠谱”的样本了。拿这样的样本训练出来的模型做召回，一叶障目，只见树木，不见森林。

就好比：一个老学究，一辈子待在象牙塔中，查古籍，访高人，能够一眼看出一件文物是“西周”的还是“东周”的。但是这样的老学究，到了潘家园，却很可能打眼，看不出一件文物是“上周”的。为什么？他一辈子只见过“老的”和“更老的”，“新的”？被象牙塔那阅人无数的门卫给屏蔽了，他压根就没见过。

拿随机采样做负样本

所以文章中描述的基本版本就是拿点击样本做正样本，拿**随机采样**做负样本。因为线上召回时，候选库里大多数的物料是与用户八杆子打不着的，随机抽样能够很好地模拟这一分布。

这个“随机抽样”，也是按照item出现的频率抽样的。在任何一个推荐系统中，“八二定律”都是不可避免的，也就是**少数热门物料占据了绝大多数的曝光与点击。**为了避免这种热点效应（召回结果都集中于少数热门物料，失去了个性化、【长尾物品】得不到曝光），我们可以：当热门物料做**正**样本时，进行降采样；当热门item做负样本的时候，要适当做过采样，增加负样本难度 -- 做in-batch**负**采样其实就是对热门商品给予更多的惩罚。所以，**在随机采样负样本时，一方面要尽可能广泛覆盖所有item，另一方面又要尽量集中于热门item。**

- 混合采样

谷歌的论文《Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations》提出了混合负采样：**batch内负采样+物料池中全局均匀采样。**batch内负采样是贴合物品出现频率的采样，节省了计算资源但是存在偏差问题，负样本不包含长尾中item；因此引入了物料池中均匀采样，引入全局分布避免batch size太小导致的分布剧变。

其实这篇论文没什么理论，更像一个实验报告，测试了不同的training batch size和index data batch size给出了一个不错的组合方式

挖掘Hard Negative增强样本

使用随机采样做负样本，也有其缺点，即与d+相比，d-与user太不匹配了。这样训练出来的模型，只能学到粗粒度上的差异，却无法感知到细微差别。就好比，一个推荐宠物的算法，能够正确做到向爱狗人士推荐狗，向爱猫人士推荐猫，但是在推荐狗的时候，无法精确感受到用户偏好上的细微差别，将各个犬种一视同仁地推出去。这样的推荐算法，用户也不会买账。

将<user,doc>对的匹配度分成三个档次：

- <user,doc>匹配度最高的，就是用户点击的商品，那是正样本。
- <user,doc>匹配度最低的，那是随机抽取的。能被一眼看穿，没难度，所谓的easy negative，达不到锻炼模型的目的。
- 所以要选取一部分匹配度适中的，能够增加模型在训练时的难度，让模型能够关注细节，这就是所谓的**hard negative。**

那么，如何选取hard-negative呢？

（1）根据业务逻辑来选取。

Airbnb在《Real-time Personalization using Embeddings for Search Ranking at Airbnb》一文中的做法，就是根据业务逻辑来选取hard negative：

- 选取与正样本同城、但是未被点击的房间作为负样本，增强了正负样本在地域上的相似性，加大了模型的学习难度。（毕竟，如果都不在同一个城市，那未免太好判断了些...）
- 增加“被房主拒绝”作为负样本（【明显的负反馈】），增强了正负样本在“匹配用户兴趣爱好”上的相似性，加大了模型的学习难度

同理，我们也可以按照商品类目、视频频道等标准来选择难负例。

（2）靠另一个模型挖掘。

当业务逻辑没有那么明显的信号时，就只能依靠模型自己来挖掘。对于业界的召回问题，我们可以用**上一版本的召回模型**筛选出“没那么相似”的<user,item>对，作为负样本，训练下一版本召回模型。怎么定义“没那么相似”呢？文章中是拿召回位置在101~500上的物料。这是因为排名太靠前那是正样本，不能用；太靠后，与随机无异，也不能用；只能取中段。

具体的做法是，我们对**线上**的数据**流**的一个batch中所有的<user,item>对都跑一遍上一版本的召回模型，计算一个batch中每对<query,document>的相似度，构成 B*B大小的矩阵。对于每一行（也就是每个query），都抽取除了对角线正样本之外的具有**最高相似度分数**的document作为难负例。

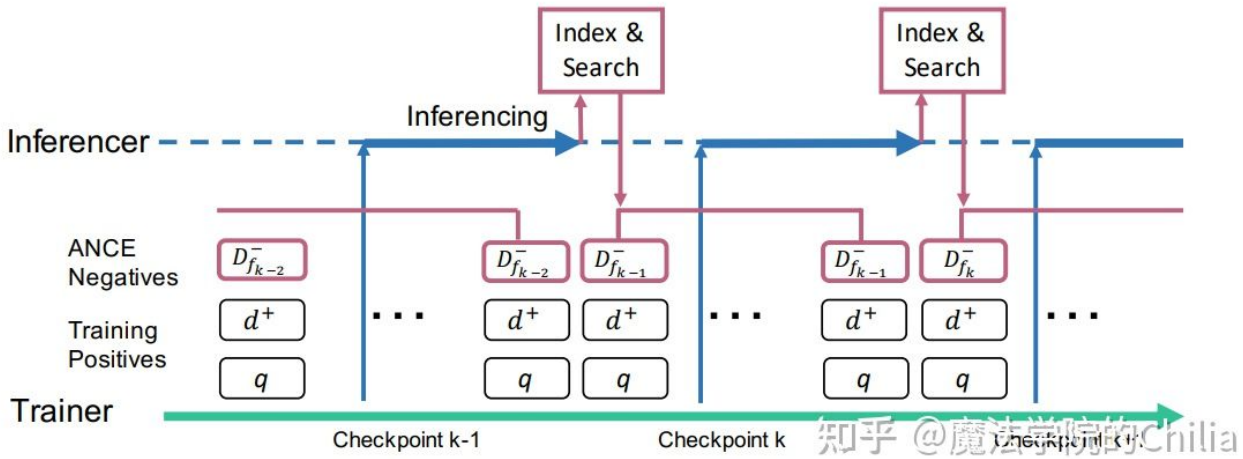
当然，我们也可以在**离线**的整体item池中去找，即对于user embedding，直接去召回item，然后取101~500位置的item。这种做法候选集太大，需要用到基于FAISS的ANN。

（3）靠模型自己来挖掘。

ANCE （Approximate Nearest Neighbor Negative Contrastive Learning for dense text retrieval, https://arxiv.org/pdf/2007.00808.pdf）

文中指出，端到端训练的深度检索模型有时效果甚至不如基于exact-match的稀疏检索模型，这是因为过多的使用简单负例（random或者in-batch负采样）没有提供很多信息量，其**梯度范数较小、收敛速度慢**。因此，需要从**全局**找到难负例。

同时，ANN查找的索引是**异步更新**的(asynchronously updated)。如果我们不用难负例采样，那么只需要在训练完成之后建立一次索引，来存储每个document的表征。但是，由于我们要进行难负例采样，所以每次训练的时候我们都需要找到根据当前模型的表征结果去找到最接近的负例。所以，每次在试图找难负例时，都使用最近的一个模型checkpoint来建立索引，然后找到和query最相近的负例作为难负例。这样找到的难负例更加informative，loss更大、梯度也更大，因此可以加速收敛。



思考：为什么要异步更新呢？更新索引需要模型的inference+存储所有document索引，虽然存索引相对效率高一些，但是inference需要在整个document库中全部经过模型的forward pass，这样的计算成本很高。训每个batch都更新一遍索引是不可接受的。所以，只需要隔一段时间用最近的checkpoint更新一下索引即可。（当然了，一种更简单的做法是用另一个训好的模型来选择难负例，但是由于这另一个模型毕竟和我们要训练的模型不同，所以不免要牺牲一些准确率。）

具体的，是用Roberta-base预训练模型来初始化双塔模型，然后先是用BM25做warm-up(用BM25做难负例采样)，之后再通过异步方法更新索引，用正在训练的模型的checkpoint进行难负例采样。

（4）人工获得难负例

将召回结果交人工审核，发现bad case，增强下一版本的训练样本。这就是一种人工寻找hard negative的方式，不太现实。

不同难度的模型融合

（1）直接把负样本放入训练集

hard negative并非要替代easy negative，而是easy negative的补充。在数量上，负样本还是以easy negative为主，文章中经验是将比例维持在easy:hard=100:1。毕竟线上召回时，库里绝大多数的物料是与用户八杆子打不着的easy negative，保证easy negative的数量优势，才能hold住模型的及格线。

（2）并行融合

把不同难度的样本单独训练一个模型，每个模型独立打分，分数计算是**多模型打分的加权和**（各模型的权重是超参，需要手工调整）。

但是线上召回时，为了能够使用FAISS，必须将多个模型生成的embedding**融合**成一个向量。做法也非常简单，将权重乘在user embedding**或**者item embedding的一侧，然后将各个模型生成的embedding**拼接起来**，再喂入FAISS。这样做，能够保证拼接向量之后的cosine值 = 与各单独向量cosine的加权和。

注意下图中，我们是将**归一化**的embedding放入了FAISS。

$$E_Q = (\alpha_1 \frac{V_{Q,1}}{\|V_{Q,1}\|}, \cdots, \alpha_n \frac{V_{Q,n}}{\|V_{Q,n}\|})$$

$$E_D = (\frac{U_{D,1}}{\|U_{D,1}\|}, \cdots, \frac{U_{Q,n}}{\|U_{Q,n}\|})$$

$$\cos(E_Q, E_D) \overset{\text{成比例}}{=} \frac{S_w(Q,D)}{\sqrt{\sum_i^n \alpha_i^2} \cdot \sqrt{n}}$$

固定的

（3）串行融合

候选item先经过easy model的初筛，再经过hard model的二次筛选，剩余结果再交给下游。这其实就是粗排的思想！

（4）课程学习

即先用in-batch negative随机采样法训练，使得第一阶段收敛之后才能进行第二阶段的课程学习。在第二阶段会有明显的指标跳变。

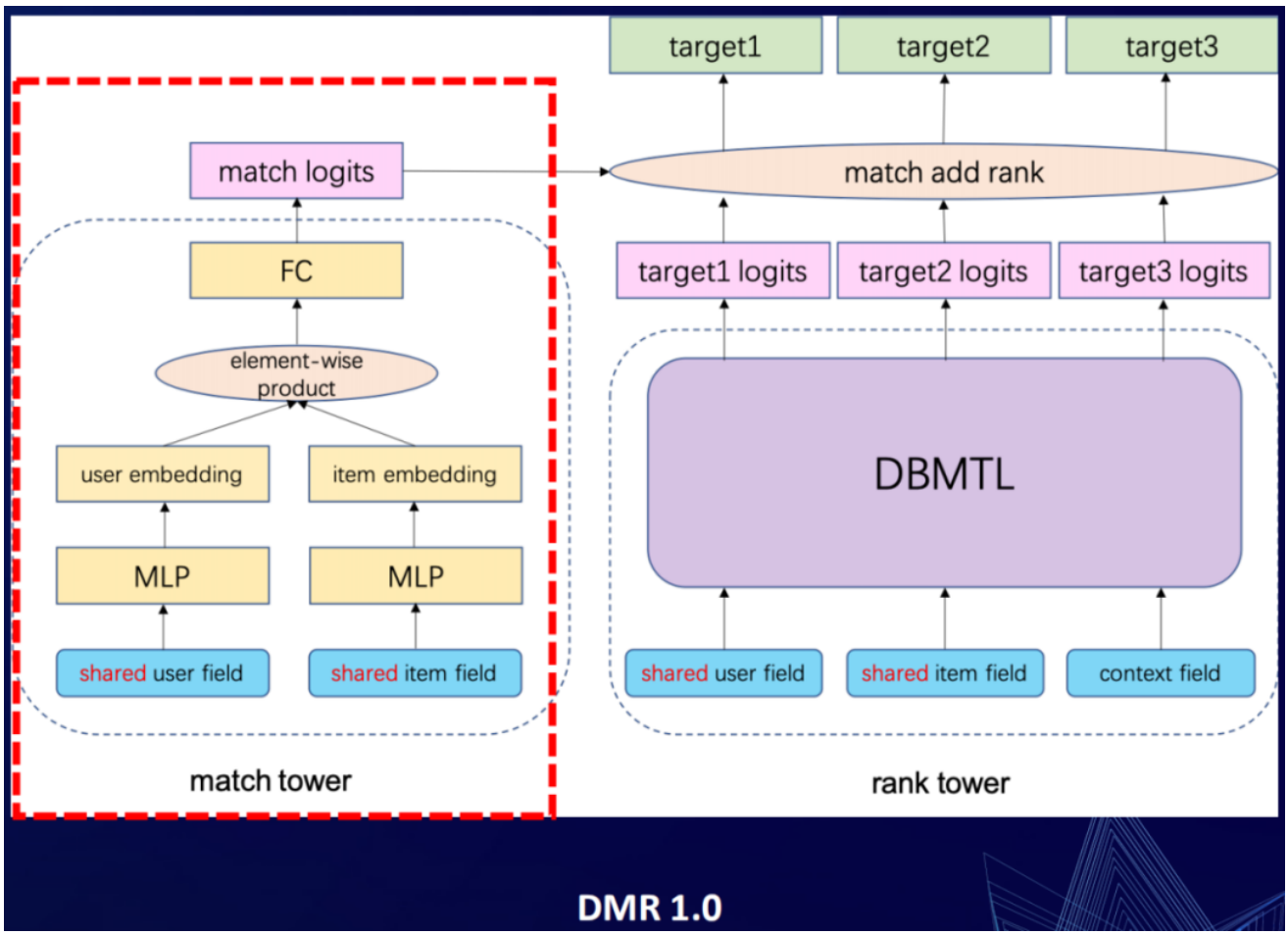
【全链路优化】

这一节切切实实cover到了我们召回工作中的痛点，就是新召回来的item，往往不受我们以往排序算法的待见

- ranker是在已有召回算法筛选过的数据的基础上训练出来的，日积月累，也强化了某种刻板印象，即<user,doc>之间的匹配模式，只有符合某个老召回描述的匹配模式，才能得到ranker的青睐，才有机会曝光给用户，才能排在容易被用户点击的好位置。
- 新的召回即使效果不错，但也未必能排出来，主要原因是排序模型是在已有召回算法筛选过的数据的基础上训练出来的。我在实际工作中也遇到了类似情况，明明召回很准，排出来的都是靠后。这是因为新召回物品所引入的<user,doc>之间的匹配模式，突破了ranker的传统认知，曝光给用户的机会就不多。即使曝光了，也会被ranker排在不讨喜的位置，不太容易被用户点击。
- 这就形成一个恶性循环，新召回的item被ranker歧视，然后曝光的机会小、排到的位置差，所以被ranker歧视的理由更加充分。形成了"强者越强，弱者越弱"得马太效应（position bias）。
- 所以优化召回后，也要对排序、融合公式等进行调整才能最大化召回的收益，但是每上一个召回都再调精排和融合公式，还要大流量看AB测试，感觉周期太长成本太高。

但是本文针对这一问题的解决方案有限：

- **将各路召回的打分作为特征，加入到训练ranker的过程中。**但是，这改变不了新召回是少数派的现实，新召回的打分作为特征在训练样本中的占比有限，发挥的作为也可能有限
- 将召回的双塔直接引入精排，作为一个辅助网络。例如阿里的DMR：模型把match的双塔和rank融合到一起，以辅助网络的形式训练【Deep Match to Rank Model for Personalized Click-Through Rate Prediction】



总而言之，各路召回之间的重叠、竞争，召回与排序之间的“相爱相杀”，将会是推荐系统内一个无解的永恒话题。

"曝光未点击"就是鸡肋

最后说一下“曝光未点击”样本。在排序中，“曝光未点击”是要被严格筛选的“真负”样本，对于排序模型的成败，至关重要。而到了召回中，“曝光未点击”样本都是妥妥的鸡肋，“食之无味，弃之可惜”

- 前面说了，“曝光未点击”不能单独拿来做负样本。
- “曝光未点击”作为hard negative增强训练集，和easy negative一起训练出一个模型，亲验线上没任何提升。Facebook的论文也指出，拿“曝光未点击”训练出来的hard model，与easy model无论是串行融合还是并行融合，都没啥效果。
- “曝光未点击”毕竟经过召回、粗排、精排，说明与用户还是比较匹配的，能否作为正样本？我也有过类似的想法，但是Facebook的论文告诉我，没啥效果。（“曝光未点击”能否作为权重较小的正样本？）

导致“曝光未点击”成为鸡肋的原因，可能是由于它的两面性：

- 它们经过上一轮召回、粗排、精排的层层筛选，推荐系统以为用户会喜欢它们。
- 而用户的行为表明了对它们的嫌弃。（当然用户“不点击"也包含很多噪声，未必代表用户不喜欢）

“曝光未点击”样本的两面性，使它们既不能成为合格的正样本，也不能成为合格的负样本。唉，“大是大非”面前，“骑墙派”总是不受欢迎的。

关于归一化和温度系数：

Youtube双塔召回《Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations》提到，双塔最上层进行**归一化**能提高最终召回效果。其实，归一化之后，我们直接求两个向量的点积就是cosine相似度了：

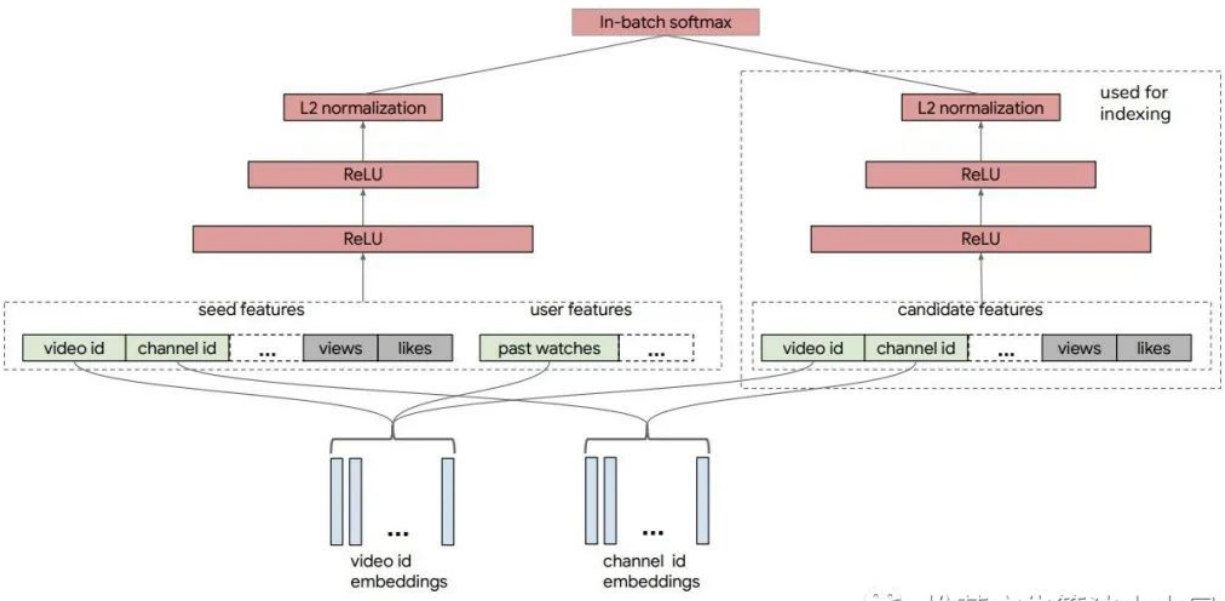


Figure 2: Illustration of the Neural Retrieval Model for YouTube.

而且归一化需要与temperature配套使用，即把计算出来的<user,item>去做带温度的softmax：

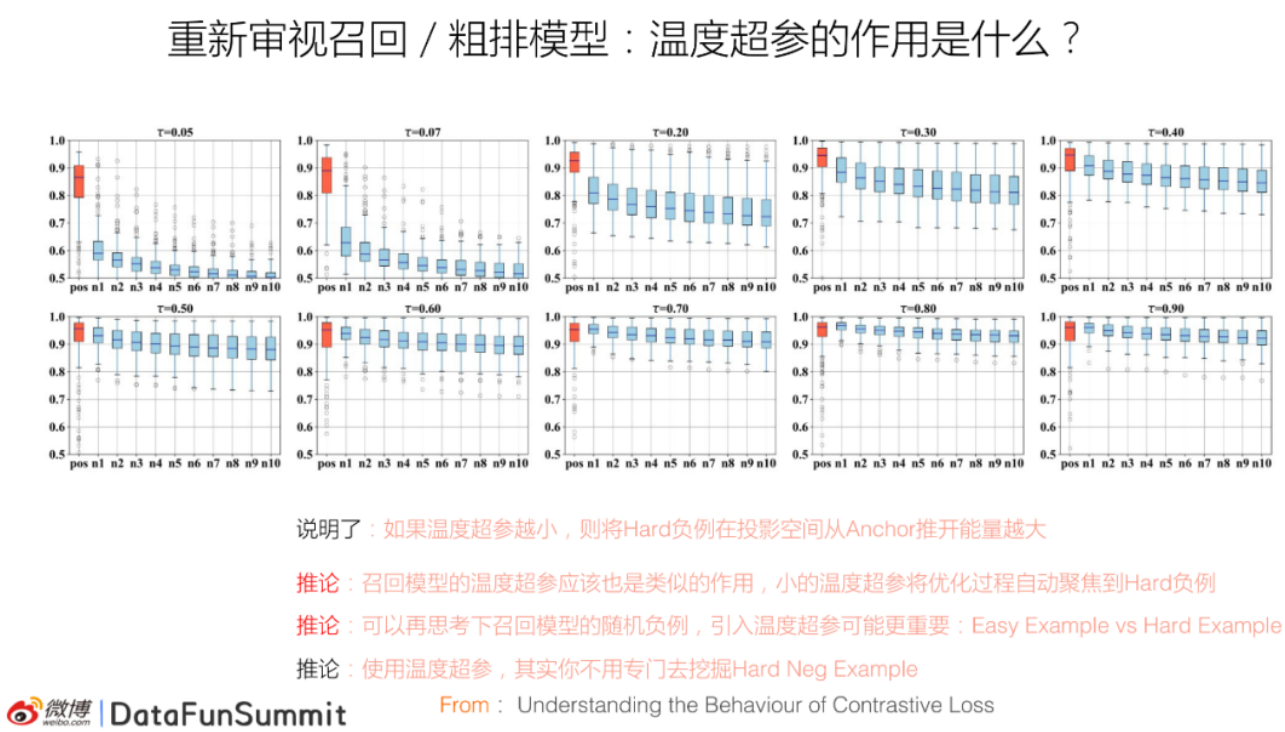
Normalization and Temperature. Empirically, we find that adding embedding normalization, i.e., $u(x,\theta) \leftarrow u(x,\theta)/\|u(x,\theta)\|_2$, $v(y,\theta) \leftarrow v(y,\theta)/\|v(y,\theta)\|_2$, improves model trainability and thus leads to better retrieval quality. In addition, a temperature τ is added to each logit to sharpen the predictions, namely,

$$s(x,y) = \langle u(x,\theta), v(y,\theta)\rangle / \tau.$$

In practice τ is a hyper-parameter tuned to maximize retrieval metrics such as recall or precision.

这个温度系数一般是一个比较小的值，加了温度之后，可以起到**sharpen softmax结果**的效果。

一个理论解释：



先说结论：小的温度超参可以让loss聚焦在**hard负例**上（因为小温度softmax实际上是将softmax的结果更加sharpen，那么，如果本来分对了的话，loss就会很小；但是如果本来模型分错了，那么小温度会让这个错误更加离谱，loss会很大，所以小温度相当于让我们先把分错的难例子给分对了。），也就是说，在大量随机选的负例中，会自动给hard负例分配更多的loss，因为这些hard负例带有更多信息，能够对我们的模型带来更多正面作用。

纵坐标代表某个锚点实例和其它不同实例的相似性得分，横坐标列出10个实例，红色的是锚点对应的正例，蓝色的代表较难区分的负例，不同的子图代表设置不同温度超参它们对应的得分变化情况。从这里可以发现：前面子图引入小的温度系数和后面比较大的温度系数，最大的区别是：如果引入的是比较小的温度系数，锚点实例和正例及hard负例的相似性距离就会被拉开，意思就是在投影空间里面，小的温度超参会把难的负例推的更远，所以这个相似性得分就拉开了，也就是说正例和hard负例区分度更明显。这就是温度超参的作用。这是个比较直观的例子。这

说明了引入温度超参而且设的温度超参越小，那么hard负例在投影空间和锚点的距离就会越远。这是图像领域里面对温度超参作用的一个解释。

那么做推荐里的召回模型，前面说过经验结论是应该引入温度超参，引入温度超参后，效果应该会有提升。怎么理解引入温度超参会对模型效果有作用呢？我的理解，应该也是上面类似的原因：对于召回模型来说，小的温度超参会将优化过程自动聚焦到hard负例上，因为hard负例带给我们的loss作用更大一些。这说明引入温度超参，能起到类似Focal Loss的作用。

如果这种解释是成立的，那么可以进一步做些推论：召回模型用的是随机负例/in-batch负例，因为引入了大量的随机负例，大量简单负例信息含量不大，但是因为数量多，所以聚集起来的loss会淹没掉hard负例的作用，那么这种情况下，引入温度超参应该更重要，因为他可以让loss更聚焦在hard负例上去。

进一步，我们可以再做更深层的推论：如果引入温度超参，其实不用花大的精力去挖掘hard负例，现在很多工作是专门挖掘hard负例的，既然温度超参的作用就是把loss聚焦在hard负例上，意味着它自动会做hard负例的选择，那么你费劲力气去挖掘hard负例就没有太大必要性。你可能需要做的是放大负例数量，然后加入温度超参，让模型自己去找hard负例。

以上是我们为什么应该在召回模型引入温度超参的一个理论解释。