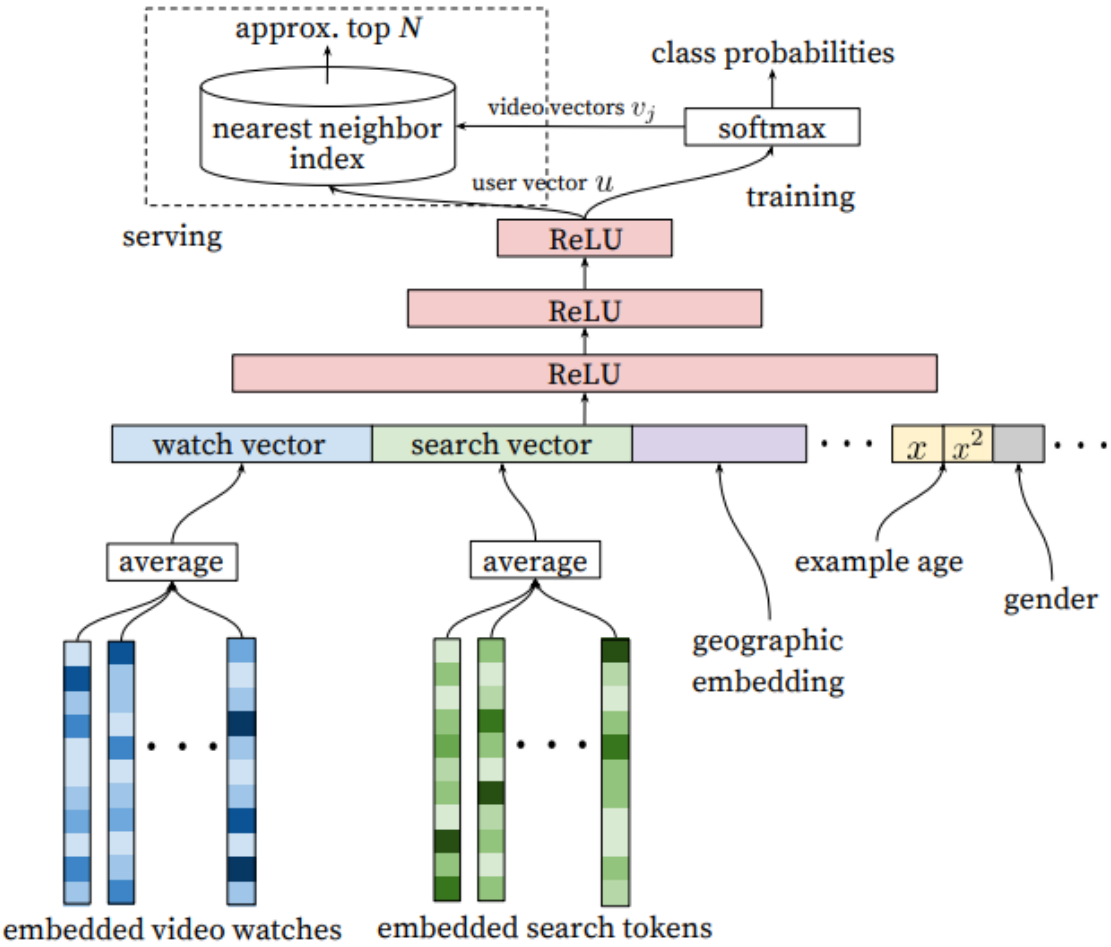
Youtube 双塔召回

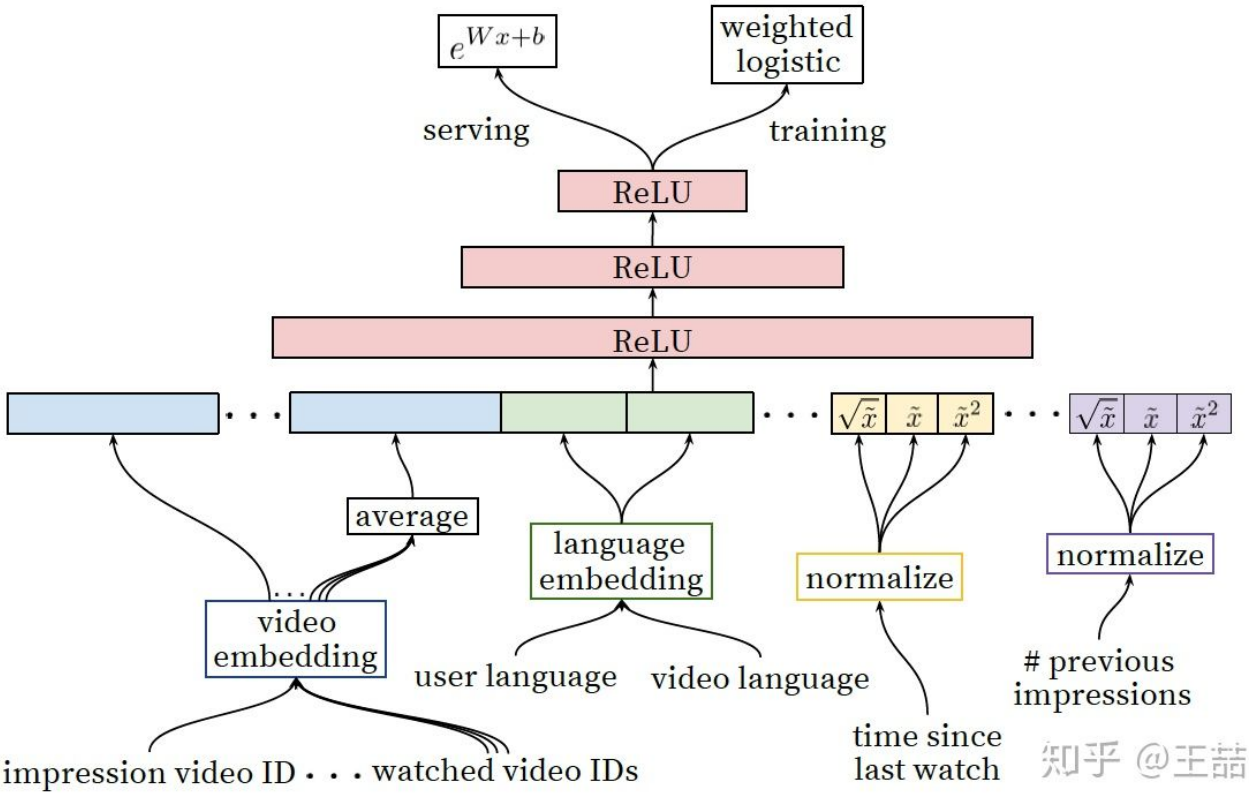
Deep Neural Network For Youtube Recommendation (Recsys 2016)

这篇文章虽然现在看起来司空见惯，但是在当时还是有很多颇具革新的思想和技巧，现在看来也值得借鉴，可谓"字字珠玑，惊为神文"。这篇文章野心很大，试图把召回和精排都囊括其中。

首先，在召回阶段，使用的特征是用户历史行为item embedding的avg-pooling，以及用户搜索query的token word2vec avg-pooling、还有用户地理位置、年龄、性别embedding（这个属于user profile信息）。三层DNN过后，我们看到了softmax函数。这里是将问题作为next-watch问题，输出是在整个candidate video上的概率分布，是一个多分类问题。但是，所有的video有亿万之巨，显然是不能生成这么大的一个softmax的。所以，我们需要负采样出B个样本，然后通过做带温度技术的sampled softmax得到概率分布。



下一部分就是精排。这里引入另一套DNN作为ranking model的目的就是引入更多描述视频、用户以及二者之间关系的**特征**，达到对候选视频集合准确排序的目的。所以这里，我们就来详细谈谈特征工程。



具体一点，从左至右的特征依次是

1. **impression video ID embedding**: 当前要计算的video的embedding
2. **watched video IDs average embedding**: 用户观看过的最后N个视频embedding的average pooling
3. **language embedding**: 用户语言的embedding和当前视频语言的embedding
4. ⭐**time since last watch**: 自上次观看同channel视频的时间
5. ⭐**#previous impressions**: 该视频已经被曝光给该用户的次数

上面五个特征中，我想重点谈谈第4个和第5个。因为这两个很好的引入了对用户行为的观察。

第4个特征用了**time since last watch**这个特征来反应用户看**同类视频**的间隔时间。从用户的角度想一想，假如我们刚看过“DOTA经典回顾”这个channel的视频，我们很大概率是会继续看这个channel的视频的，那么该特征就很好的捕捉到了这一用户行为。

第5个特征**#previous impressions**则一定程度上引入了exploration的思想，避免同一个视频持续对同一用户进行无效曝光。尽量增加用户没看过的新视频的曝光可能性。

思考几个问题：

1.Youtube的用户对新视频有偏好，那么在模型构建的过程中如何引入这个feature?

模型引入了“Example Age”这个特征，会把Days since Upload作为这个example age加入模型中。

2.在对训练集的预处理过程中，Youtube没有采用原始的用户日志，而是对每个用户**提取等数量的训练样本**，这是为什么?

原文的解答是这样的，

> Another key insight that improved live metrics was to generate a fixed number of training examples per user, weighting our users equally in the loss function. This prevented a small cohort of **highly active users from dominating the loss**.

理由很简单，这是为了减少高度活跃用户对于loss的过度影响。

3.在进行video embedding的时候，为什么要直接把大量长尾的video直接用0向量代替？

这又是一次工程和算法的trade-off，把大量长尾的video截断掉，主要还是为了节省online serving中宝贵的内存资源。当然从模型角度讲，低频video的embedding的准确性不佳是另一个“截断掉也不那么可惜”的理由。

当然了，把长尾item用0向量代替并不是最好的方法，实际对于大规模稀疏ID类特征可以用HashBucket去映射，把长尾的ID当成OOV来处理，映射到一个”公共溢出区“。

4.针对某些特征，比如#previous impressions，为什么要进行开方和平方处理后，当作三个特征输入模型？

这是很简单有效的工程经验，引入了特征的非线性。从YouTube这篇文章的效果反馈来看，提升了其模型的离线准确度。当然了，这种方法现在已经很少使用，取而代之的是把连续特征离散化分桶，这样更能够提供非线性，还能够和其他特征进行交互。

Sampling Bias-Corrected Neural Modeling for Large Corpus Item Recommendations (Recsys 2019)

1. 简介

大规模推荐系统一般分为两个阶段，即召回和排序阶段。本文主要关注推荐系统的召回阶段。召回阶段的一个常用方法就是分解user-item矩阵。但是，这样只能够把握住user-item交互信息；而我们知道，增加content信息可以缓解数据稀疏的问题（因为虽然交互数据少，但是我知道item/user的内容本身信息呀），提升泛化能力。之前我们讲过，Factorization machine其实就是模拟分解一个带有content信息的user-item交互矩阵的过程；深度神经网络则更自然地能把content信息也统统加进来。

问题描述：给定{用户(user)，上下文(context)，物品(item)}的三元组，召回模型通用的方法是：

- 1）分别计算{user,context} 和 {item} 的向量表示；
- 2）利用一个简单的打分函数（例如点积）来计算二者的匹配得分。

其中，"context"通常是有动态性质的变量，例如一天里的时间、用户的设备等等。这种基于表示学习的方法通常面临两个方面的挑战：

- 1）工业级推荐系统的item集通常很大；
- 2）从用户反馈中收集的训练数据通常对于长尾物品很**【稀疏】**，导致对长尾内容的预测**【过拟合】**（只有记忆性没有泛化性），长尾物品的ID embedding由于数据量很少所以学的很差。那么怎么办呢？一个办法就是，加入content 特征。之前讲类别型特征embedding的时候我们就说过，加入一些side feature可以提升ID embedding的泛化能力。对于【cold-start】问题，我们虽然没有用户的交互特征，但是好在我们有content特征，应当好好利用。

工业界现有的推荐系统都需要从一个超大规模的候选集中拉取item进行打分排序。解决数据稀疏和大规模候选集分布的一种通用做法是从item的内容特征(context feature)中学习出item的**稠密**embedding表示。这里很自然地就想到了工业界大名鼎鼎且应用广泛的双塔神经网络结构，其中的一塔就是从丰富的item content feature中学习到item的embedding表示。双塔网络结构如下所示：

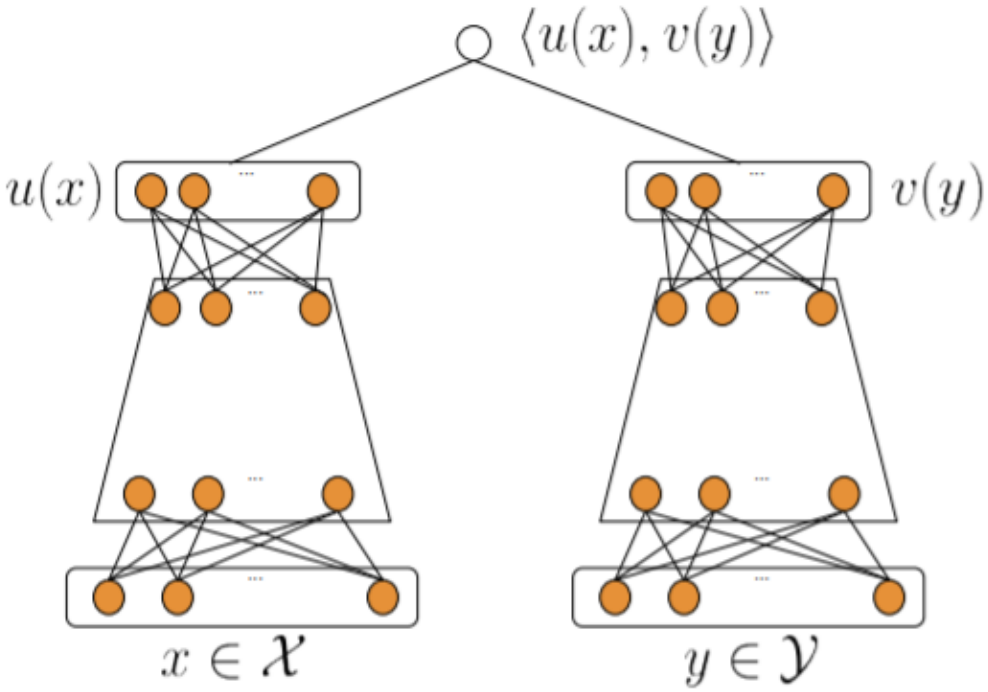


Figure 1: A two-tower DNN model for learning query and candidate representations.

左侧：{user,context}塔，即query；右侧：{item}塔

上图中双塔模型两侧分别对 {user,context} 和 {item} 进行建模，并在最后一层计算二者的内积。模型训练好后用图中的两个向量 $\boldsymbol{u(x)}$和 $\boldsymbol{v(y)}$求点积，召回top-k个点积最大的item。其中，左侧user塔的user embedding需要在请求来的时候实时计算；右侧item塔训练的时候预计算好，然后灌入一个向量检索工具中，首先建立索引，然后转化为一个向量检索问题，这方面基本做推荐的各家大公司都有自己的开源工具，比如faiss（Facebook AI开源的向量相似度计算引擎，浅析Faiss在推荐系统中的应用及原理），annoy等。

YouTube将视频召回看做一个极端的**多分类**问题，多分类问题中最常使用的激活函数就是softmax，但是要知道YouTube上视频素材库中的视频数量是巨大的，当类别数量特别大时，使用softmax来训练模型是比较耗时的（因为分母需要计算user端和*全部*tem embedding的相似度指数求和）。所以一种常用的方法就是进行负采样(类似skip-gram中的【negative-sampling】)，工业界目前采用流式训练双塔结构一般是通过随机**mini-batch**内负采样的方式来优化损失函数。这种训练方式存在的一个显著问题就是in-batch loss会因为随机采样bias而导致模型效果不好，即：对于热门物品来说，**由于采样到的概率非常高，当作负样本的次数也会相应变多，热门物品会被"过度惩罚"。而在大多数推荐场景中有很明显的热点效应，对于这些热门item的过度打压会使得模型倾向于推荐一些冷门的item，从而影响线上表现。**

流式训练，训练数据是以滚动的方式输入到网络模型中的，我们无法维护一个动态且占高内存的词表及item频次信息。如果无法拿到item出现的频次，就不能准确的进行负采样操作。

2. 负采样方法

2.1 采样概率修正

整个召回系统采用的是双塔结构，即分别构建user侧的Embedding和item侧的Embedding，两个塔的输出就是各自的embedding向量，最终模型的输出为两个Embedding内积后的结果，即：

$$s(x,y) =< u(x,\theta), v(y,\theta) >$$

假设有T条训练样本，模型的目标是从这些训练样本中学习模型参数 θ 。数据集定义如下：

$$\mathcal{T} := \{(x_i, y_i, r_i)\}_{i=1}^{T},$$

其中， x_i 表示user特征和context特征的融合embedding, y_i 表示item的特征embedding, r_i 表示每条样本的回报，比如文章的阅读时长、视频的播放比例等，损失函数中用 r_i 加权。

这篇论文**将视频召回看做一个多分类的问题**，通常各个类别的权重都为1，但是在该场景中，论文中引入了一个用户偏好的权重 r_i （例如用户观看某视频的播放比例）。给定一个用户u，需要从M个物品的候选集中选择要推荐的物品。这是一个多分类问题，将模型的输出经过一个softmax函数之后得到具体对应的类别概率，多分类softmax函数定义如下：

预测点击 y 的概率

$$\mathcal{P}(y|x;\theta) = \frac{e^{s(x,y)}}{\sum_{j\in[M]} e^{s(x,y_j)}}. \quad (1)$$

x 和 y 的相似度

x 和其它的有 y 的相似度

分母需要计算M个物品，当M很大时根本不可行！

损失函数采用加权对数似然函数的形式，具体如下：

$$L_T(\theta) := -\frac{1}{T}\sum_{i\in[T]} r_i \cdot \log(\mathcal{P}(y_i|x_i;\theta)).$$

因为在实际工业应用中，我们考虑的是流式数据。和离线数据不同，流式数据没有一个固定的、能统计数据分布的数据集。所以，一种通用的做法是通过mini-batch的方式来优化损失函数：假设一个包含B条数据的mini-batch，那么对于任意一条数据，softmax计算公式如下：

$$\mathcal{P}_B(y_i|x_i;\theta) = \frac{e^{s(x_i,y_i)}}{\sum_{j\in[B]} e^{s(x_i,y_j)}}.$$

这种做法相当于**把一个batch中此条数据之外物品当作负样本**。但是这种做法存在的缺点就是会因为随机采样偏差而导致模型效果不好：对于热门物品来说，由于采样到的概率非常高，当作负样本的次数也会相应变多，热门物品会被"过度惩罚"。所以论文对user和item的embedding向量计算得到的内积进行了修正，即：

$$s^c(x_i,y_j) = s(x_i,y_j) - \log(p_j).$$

上式中, p_j 代表物品 j 在一个mini-batch中被采样到的概率。那么 p_j 怎么估计呢？这在下一节会详细介绍。基于此修正的内积，得到修正后的softmax函数的输出：

$$\mathcal{P}_B^c(y_i|x_i;\theta) = \frac{e^{s^c(x_i,y_i)}}{e^{s^c(x_i,y_i)} + \sum_{j\in[B],j\neq i} e^{s^c(x_i,y_j)}}.$$

以及修正后的损失函数如下所示：

$$L_B(\theta) := -\frac{1}{B}\sum_{i\in[B]} r_i \cdot \log(\mathcal{P}_B^c(y_i|x_i;\theta)), \qquad (4)$$

然后利用优化器SGD进行参数的更新。具体的模型训练算法如下所示：

Algorithm 1 Training Algorithm

Input: Two parameterized embedding functions $u(\cdot,\theta), v(\cdot,\theta)$ where each one maps input features to an embedding space through a neural network. Learning rate γ (fixed or adaptive).
repeat
  Sample or receive a batch of training data $\{(x_i, y_i, r_i)\}_{i=1}^{B}$ from a stream.
  Obtain the estimated sampling probability p_i of each y_i from Algorithm 2.
  Construct loss $L_B(\theta)$ according to (4).
  $\theta \leftarrow \theta - \gamma \cdot \nabla L_B(\theta)$.
until stopping criterion

2.2 Streaming Frequency Estimation

$$p_j$$

p_j 怎么估计呢？文中提出的估计方法的核心思想在于，通过采样频率来估计采样概率。例如，如果一个item平均50个step被采样到1次，那么它的采样概率p = 0.02。

由于YouTube中采用流式训练，因此不断会有新物品出现，那么使用固定长度的词表不太合适，因此采用哈希的方式来对物品的采样概率

$$p_j$$

p_j 进行更新。

具体来说，首先定义一个哈希函数，把所有视频id都映射到[0,1,...H]之间。这里哈希一下是因为视频是动态的，随时都可能有新的视频进入系统，所以用哈希函数映射一下固定住视频库大小。同时使用两个长度为H的数组A和B，通过哈希来得到给定的物品j在数组A和B中的下标。

- A[h(y)]: 表示物品y上一次被采样的step；
- B[h(y)]: 表示物品j平均多少个step被采样一次，即采样频率。其倒数就是预估被采样到的概率。

估计采样概率的算法如下：

Algorithm 2 Streaming Frequency Estimation

Input: Learning rate α. Arrays A and B with size H. Hash function h with output space $[H]$.
(Training)
For steps $t = 1, 2, \ldots$.
  Sample a batch of items $\mathcal{B}$. For each $y \in \mathcal{B}$, do
    $B[h(y)] \leftarrow (1-\alpha) \cdot B[h(y)] + \alpha \cdot (t - A[h(y)])$.
    $A[h(y)] \leftarrow t$.
Until stopping criterion
(Inference)
  For any item y, sampling probability $\hat{p} = 1/B[h(y)]$.

这里还有一个问题，就是只要有哈希，就会有冲突。冲突的情况会使不同的id聚在同一个桶，导致采样概率预估过高。这里的改进方案是使用Multiple Hashings。即使用多个哈希函数和A、B数组。在线推理时，使用最大的一个

$$B_i[h(y)]$$

B_i[h(y)] 去计算采样概率。

当然论文里还给出来了"Streaming Frequency Estimation"算法的分布式计算方法，这里把用于估计流式数据中，每个batch下item被采样的概率的算法引申到深度学习的分布式训练中，真的是把这个算法运用到工业中落地的细节也讲到了，想详细了解这一块的知识可以读原论文。//TODO: Tensorflow的分布式训练

2.3 一些Tricks

1）「归一化」：经验表明，对两侧输出的Embedding进行归一化效果更好，即：

$$u(x,\theta) \leftarrow \frac{u(x,\theta)}{||u(x,\theta)||_2}$$
$$v(x,\theta) \leftarrow \frac{v(x,\theta)}{||v(x,\theta)||_2}$$

2）「内积除以温度系数 τ 」：

$$s(x,y) = \frac{s(x,y)}{\tau}$$

除以温度系数 τ 的效果是**把softmax的结果变得更加明显（sharpen）**，通过对τ的微调可以使召回率或精确率变的更好。由于我们是对一个正样本和若干负样本求softmax，所以需要这个温度系数来让softmax的结果更加明显。在实际应用中，如果不加这个温度系数，就会导致不收敛。

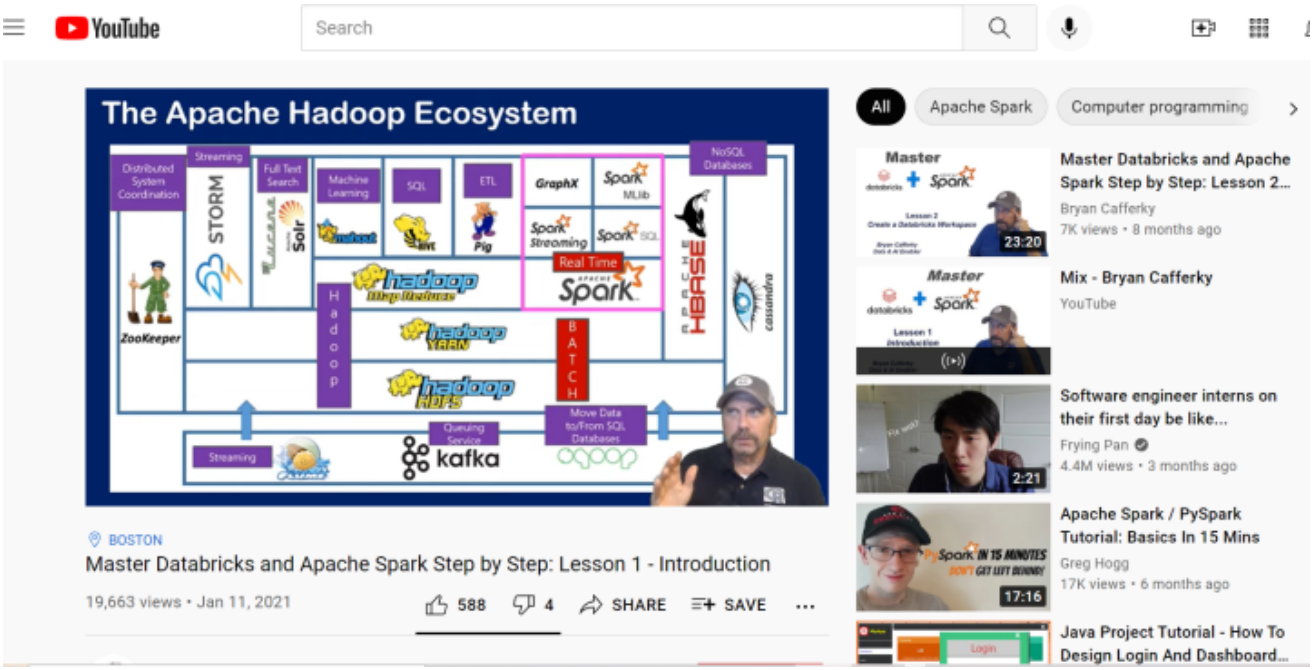
对于温度系数，和之前讲AutoDis时提到的温度系数一样：

> 对于温度系数，当其接近于0时，得到的分桶概率分布接近于one-hot，当其接近于无穷时，得到的分桶概率分布近似于均匀分布。这种方式也称为softargmax。

3. 模型结构与线上部署

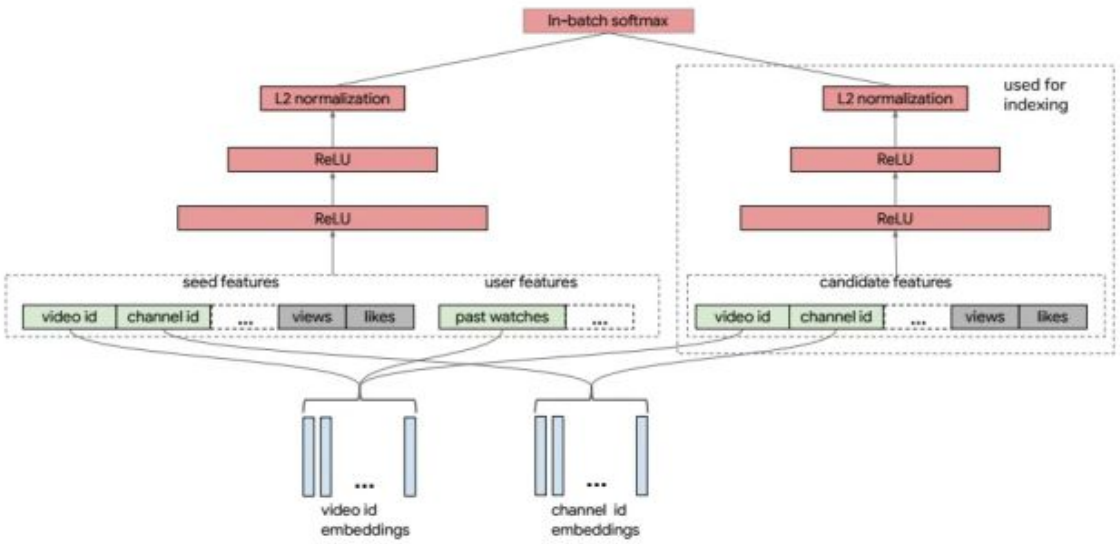
本文针对的业务场景是，在用户观看某个视频的时候旁边的推荐视频列表。其中，我们正在观看的这个视频叫做seed video。

例如：



3.1 模型结构

seed video是表示用户当前兴趣的强特征，所以我们要对seed video特征进行更好的表示，然后再融合进用户的历史行为信息。整个召回的模型网络如下图所示：



将模型分为两个塔，query塔（包含user features 和 context feature）和item塔，模型使用大量的seed video特征和用户观看历史来训练模型。seed video features 是指用户正在观看的视频所具有的属性，某种程度上表征了用户当前的兴趣。

训练目标：还是预测是否点击。但是，这里一个优化的方式是对损失函数引入用户satisfaction加权，例如以user watch time作为权重，当用户点击但是很快就关掉的时候，权重$r_i \approx 0$; 如果用户把整个视频都看完了，$r_i = 1$.

3.2 Feature Embedding

1) 视频特征有诸如视频id、频道id之类的类别型特征，也有连续型特征。类别特征分为单值类别特征（如 video id）和多值类别特征（如 video category）。单值类别特征直接用embedding table转化为embedding，对于多值类别特征，采用对其多值embedding加权求和的方式得到最终的embedding。

2）用户特征主要是基于用户的历史观看记录来捕获用户的兴趣，比如说，用户最近看过k个视频的embedding的平均值来刻画用户兴趣。（我认为，在这里只简单用了平均值而没有用更复杂的方法的原因是，这是召回而不是排序，不能使用复杂度很高的算法。）

对于ID型的类别特征，embedding在模型的两侧是**共享**的。比如，视频id的embedding两个塔是一样的，这样是为了保证两个塔在一个空间内，同时减低存储压力。作者也尝试使用不共享的embedding，并没有发现明显提升。

这里作者还提到【解决OOV】的方法：随机哈希到一个桶中，并且为每个桶都学习一个embedding。

3.3 在线服务

模型基于Tensorflow实现，并且进行了分布式实现。同时，YouTube 每天都会生成新的训练数据集，模型也会以天为单位进行更新，这样模型便能适应最新的数据分布变化。

infer阶段就跟普通的DNN召回类似，将全量的item embedding离线计算好，然后线上实时生成query塔的embedding，通过faiss等工具进行topK查询。

4. 总结

最后，借此机会来回顾YouTube的三篇重要的实践论文：

（1）首先是YouTube的深度学习开山之作《Deep Neural Networks for YouTube Recommendations》，这是深度学习在推荐领域的首次工业应用，这篇文章野心很大，将召回和排序都囊括进整篇论文。

（2）其次是这篇YouTube的双塔召回，进一步优化解决召回的冷启动、长尾等问题。

（3）最后是MMoE的应用文章《Recommending What Video to Watch Next: A Multitask Ranking System》，该论文主要聚焦于大规模视频推荐中的排序阶段，介绍一些比较实在的经验和教训，涉及到对多个目标进行建模和解决 Selection Bias这两个排序系统的关键点。