万变不离其宗：用统一框架理解向量化召回

Original 石塔西 推荐道 Today

收录于话题

#推荐算法 13 #搜索算法 2 #深度学习 9 #召回算法 4 #图神经网络 6

前言

常读我的文章的同学会注意到，我一直强调、推崇，不要孤立地学习算法，而是要梳理算法的脉络+框架，唯有如此，才能真正融会贯通，变纸面上的算法为你的算法，而不是狗熊掰棒子，被层出不穷的新文章、新算法搞得疲于奔命。

之前，我在《推荐算法的"五环之歌"》梳理了主流排序算法常见套路：

- 特征都ID化。类别特征天然是ID型，而实数特征需要经过分桶转化。
- 每个ID特征经过Embedding变成一个向量，以扩展其内涵。
- 属于一个Field的各Feature Embedding通过Pooling压缩成一个向量，以减少DNN的规模
- 多个Field Embedding拼接在一起，喂入DNN
- DNN通过多层Fully Connection Layer (FC)完成特征之间的高阶交叉，增强模型的扩展能力。
- 最后一层FC的输出，就是最终的logit，与label（e.g., 是否点击？是否转化）计算binary cross-entropy loss。

相比于排序那直白的套路，召回算法，品类众多而形态迥异，看似很难找出共通点。如今比较流行的召回算法，比如：item2vec、DeepWalk、Youtube的召回算法、Airbnb的召回算法、FM召回、DSSM、双塔模型、百度的孪生网络、阿里的EGES、Pinterest的PinSAGE、腾讯的RALM和GraphTR、.....

- 从召回方式上分，有的直接给user找他可能喜欢的item（user-to-item，简称u2i）；有的是拿用户喜欢的item找相似item（item-to-item，简称i2i）；有的是给user查找相似user，再把相似user喜欢的item推出去（user-to-user-to-item，简称u2u2i）
- 从算法实现上分，有的来自"前深度学习"时代，老当益壮；有的基于深度学习，正当红；有的基于图算法，未来可期（其实基于图的，又可细分为游走类和卷积类）。
- 从优化目标上分，有的属于一个越大规模的多分类问题，优化softmax loss；有的基于Learning-To-Rank(LTR)，优化的是hinge loss或BPR loss

但是，如果我告诉你，以上这些召回算法，其实都可以被一个统一的算法框架所囊括，惊不惊奇、意不意外？本文就介绍我归纳总结的**NFEP（Near, Far, Embedding, Pairwie-loss）**框架，**系统化地理解向量化召**

回算法。在详细介绍之前，我首先需要强调，这么做的目的，并非要将本来不相干的算法“削足适履”硬塞进一个框架里，哗众取宠，而是有着两方面的重要意义：

- 一是为了开篇所说的“**融会贯通**”。借助NFEP，你学习的不再是若干孤立的算法，而是一个算法体系，不仅能加深对现有算法的理解，还能轻松应对未来出现的新算法；
- 二是为了“**取长补短**”。大多数召回算法，只是在NFEP的某个维度上进行了创新，而在其他维度上的做法未必是最优的。我们在技术选型时，没必要照搬某个算法的全部，而是借助NFEP梳理的脉络，博采多家算法之所长，取长补短，组成最适合你的业务场景、数据环境的算法。

接下来，我首先介绍NFEP框架，然后逐一介绍如何从NFEP的框架视角来理解Airbnb召回、Youtube召回、Facebook EBR、Pinterest的PinSAGE、微信GraphTR、FM召回这几种典型的召回算法。

NFEP：理解向量化召回的统一框架

向量化召回简介

NFEP框架关注的是“向量化召回”算法，也就是将召回建模成在向量空间内的近邻搜索问题。传统的ItemCF/UserCF那种基于统计的召回方式，和阿里TDM那种基于树模型层次化划分搜索空间的召回算法，不在讨论范围之内。

假设向量化召回，是拿X概念下的某个x，在向量空间中搜索Y概念下与之最近的y。其serving的套路就是

- 离线时，将几百万、上千万的y，通过模型获得它们的embedding，将这些y embedding灌入FAISS并建立索引
- 在线时，拿请求中的x，或提取、或生成x embedding，在FAISS中查找最近的y embedding，将对应的y作为召回结果返回

为了达成以上目标，我们在训练的时候，需要考虑四个问题：（1）如何定义X/Y两概念之间的“距离近”？（2）如何举反例，即如何定义X/Y之间的“距离远”？（3）如何获取embedding?（4）如何定义loss来优化?

这4个问题，对应着NFEP框架的4个维度，接下来将会逐一详细分析。

Near：如何定义“近”?

这取决于不同的召回方式

- i2i召回：x,y都是item，我们认为同一个用户在同一个session交互过的两个item在向量空间是相近的，体现两个item之间的“相似性”。
- u2i召回：x是user，y是item。一个用户与其交互(e.g., 点击、观看、购买)过的item应该是相近的，体现user与item之间的“匹配性”。
- u2u召回：x,y都是user。比如使用孪生网络，则x是user一半的交互历史，y是同一用户另一半交互历史，二者在向量空间应该是相近的，体现“同一性”。

无论哪种召回方式，为了能够与FAISS兼容，我们都拿x embedding和y embedding之间的“点积”或"cosine"来衡量距离。显然，点积或cosine越大，代表x与y在向量空间越接近。

Far：如何定义“远”？

其实就是举反例。举出的<x, y_->反例，要能够让模型见识到形形色色、五花八门、不同角度的“x, y_-之间差异性”，达到让模型“**开眼界，见世面**”的目的。特别是在训练u2i召回模型时，一个非常重要的原则就是，**千万不能（只）拿“曝光未点击”做负样本**。否则，正负样本都来自“曝光”样本，都是与user比较匹配的item，而在上百万的候选item中，绝大部分item都是与user兴趣“八杆子打不着”的。这种训练数据与预测数据之间的bias，将导致召回模型上线后“水土不服”。具体原理解释，请参考我的另一篇文章《负样本为王》。

为了达到以上目标，获得y_-最重要的方式就是在整个y的候选集中随机采样：

- 有同学担心，随机采样得到的y_-有可能与x是相近的。不排除这种可能性，但是在一个实际的推荐系统中，候选的y一般是成百上千万，而每次随机采样的y_-不会超过100，这种false negative的概率极低。
- 随机采样并非uniform sampling，那样会导致热门item霸占召回结果，从而失去个性化。因此在采样时，需要打压热门item。比较有效的一种方式就是学习word2vec中打压高频词的方法，降低热门item成为正样本的概率，提升热门item成为负样本的概率。具体公式细节，见的我的知乎回答《推荐系统传统召回是怎么实现热门item的打压?》。

但是，只通过随机采样获得y_-也有问题，就是会导致模型的精度不足。假如，你要训练一个“相似图片召回”算法。

- 当x是一只狗时，y_+是另外一只狗。y_-在所有动物图片中随机采样得到，大概率是到猫、大象、乌鸦、海豚、...。这些随机负样本，对于让模型“开眼界，见世面”十分重要，能够让模型快速“去伪存真”。
- 但是猫、大象、乌鸦、海豚、...这样随机的负样本，与正样本相比相差太大，使得模型只观察到粗粒度就足够了，没有动力去注意细节，所以这些负样本被称为easy negative。

- 为了能给模型增加维度，迫使其关注细节，我们需要让其见识一些hard negative，比如狼、狐狸、...这种与x 、 y_+还有几分相似的负样本

不同的算法，采取不同的方式获得hard negative，在下文中将会详细分析。

Embedding：如何生成向量?

用哪些特征学出embedding?

- 有的算法只使用UserId/DocId
- 有的算法除此之外，还使用了画像、交互历史等side information
- 图卷积算法还使用了user节点、item节点在图上的连接关系

通过什么样的模型学习出embedding?

- 只使用ID特征的算法，模型就只有一个embedding矩阵，通过id去矩阵相应行提取embedding
- 有的模型，将特征（id+side information）喂入DNN，逐层让特征充分交互，DNN最后一层的输出就是我们需要的embedding
- 基于图卷积的模型中，目标节点(user或item)的embedding，是由其邻居节点的embedding，加上节点本身信息，聚合而成。一来，图上的节点不仅有user/item，还可以包括性别、年龄、职业等属性节点；二来，邻居节点也有自己的邻居。图的这种性质，使得节点embedding能够利用的信息更加广泛，并且兼具本地与全局视角。

召回模型的特点：解耦

u2i召回，与排序，虽然都是建模user与item的匹配（match）关系，但是在样本、特征、模型上都有显著不同。在之前的文章中，我详细论述了二者在样本选择上的区别，这一节将论述二者在特征、模型上的区别。简言之，就是：排序鼓励交叉，召回要求解耦。

排序鼓励交叉

- 特征上，排序除了利用user feature(包括context)、item feature，最重要还使用了大量的交叉统计特征，比如“user tag与item tag的重合度”。这类交叉统计特征是衡量“user与item匹配性”的最强信号，但是也将user feature与item feature紧密耦合在一起。
- 模型上，排序一般将user feature、item feature、交叉统计特征拼接成一个大向量，喂入DNN，让三类特征通过多层全连接层（Fully Connection, FC）进行充分交叉。从第一层FC之后，你就已经无法分辨，

FC的输出中哪些属于user信息？哪些属于item信息？

召回要求解耦

排序之所以允许、鼓励交叉，还是因为它的候选集比较小，最多不过几千个。**换成召回那样，要面对百万、千万级别的海量候选item，如果让每个user与每个候选item都计算交叉统计特征，都过一遍DNN那样的复杂操作，是无论如何也无法满足线上的实时性要求的。**所以，召回要求解耦、隔离user与item特征。

- 特征上，尽管信号强，但是**召回不允许使用"交叉统计特征"。**（放弃这么强的信号，的确可惜。如何在不使用交叉统计特征的情况下，仍然达到使用了它们的效果？有一种方法是使用蒸馏，详情见《Privileged Features Distillation at Taobao Recommendations》）
- 模型上，**禁止user feature与item feature出现DNN那样的多层交叉，二者必须独立发展**，i.e., user子模型，利用user特征，生成user embedding；item子模型，利用item特征，生成item embedding。唯一一次user与item的交叉，只允许出现在最后拿user embedding与item embedding做点积计算匹配得分的时候。

只有这样，才能允许我们

- 离线时，在user未知的情况下，独立生成item embedding灌入faiss；
- 在线时，能独立生成user embedding，避免与每个候选item进行"计算交叉特征"和"通过DNN"这样复杂耗时的操作

Pairwise-loss：如何成对优化?

排序阶段经常遵循"CTR预估"的方式

- 样本上，<$user, item_+, 1$>和<$user, item_-, 0$>是两条样本
- loss上使用binary cross-entropy这样的pointwise loss

能够这么做的前提是，其中的<$user, item_-, 0$>是"曝光过但未点击"的"真负"样本，label的准确性允许我们使用pointwise loss追求"绝对准确性"。

但是在召回场景下，以上前提并不成立。以常见的u2i召回为例，绝大多数item从未给user曝光过，我们再从中随机采样一部分作为负样本，这个negative label是存在噪声的。在这种情况下，再照搬排序使用binary cross-entropy loss追求"预估值"与"label"之间的"绝对准确性"，就有点强人所难了。所以，召回算法往往采用Pairwise LearningToRank（LTR），建模"**排序的相对准确性**":

- 样本往往是<$user, item_+, item_-$>的三元组形式

- 模型的优化目标是，**针对同一个user，$item_+$与他的匹配程度，要远远高于，$item_-$与他的匹配程度。**所以Loss中没有label，而存在“<$user, item_+$>的匹配分”与“<$user, item_-$>的匹配分”相互比较的形式。

为了实现Pairwise LTR，有几种Loss可供选择。

一种是sampled softmax loss。

- 这种loss将召回看成一个超大规模的多分类问题，优化的目标是使，user选中item+的概率最高。
- user选中item+的概率= $\frac{exp(v_u \cdot v_{i_+})}{\sum_{|I|} exp(v_u \cdot v_i)}$，其中$v_u$是user embedding，$v_i$代表item embedding，$|I|$代表整个item候选集。
- 为使以上概率达到最大，要求分子，即user与item+的匹配度，尽可能大；而分母，即user与除item+之外的所有item的匹配度之和，尽可能小。体现出上文所说的“**不与label比较，而是匹配得分相互比较**”的特点。
- 但是，由于计算分配牵扯到整个候选item集合$|I|$，计算量大到不现实。所以实际优化的是sampled softmax loss，即从$|I|$中随机采样若干$item_-$，近似代替计算完整的分母。
- $L_{SampledSoftmax} = \sum_{(u,i_+)} log(1 + exp(-v_u \cdot v_{i_+})) + \sum_{(u,i_-)} log(1 + exp(v_u \cdot v_{i_-}))$

另一种loss是margin hinge loss，

- 优化目标是：user与正样本item的匹配程度，要比，user与负样本item的匹配程度，高出一定的阈值。
- 即，L_{hinge}=$\sum_{i}^{n} max(0, margin - user \cdot item_+ + user \cdot item_-)$

因为margin hinge loss多出一个超参margin需要调节，因此我主要使用如下的BPR Loss。

- 其思想是计算"给user召回时，将item+排在item-前面的概率"，
  $p_{CorrectOrder} = sigmoid(user \cdot item_+ - user \cdot item_-)$
- 因为<user, item+, item->的ground-truth label永远是1，所以将$p_{CorrectOrder}$喂入binary cross-entropy loss的公式，就有$L_{BPR} = -log(p_{CorrectOrder})$=$log(1 + exp(user \cdot item_- - user \cdot item_+))$

注意，为了方便行文，以上公式都是针对u2i召回的举例，但是u2u, i2i召回也具备类似的公式。

用NFEP理解典型向量化召回算法

本节将从NFEP框架的视角，来理解几种主流、经典的召回算法，看看这些算法是在哪些维度上进行了创新，存在哪些内在联系。

Airbnb召回算法

《Real-time Personalization using Embeddings for Search Ranking at Airbnb》是一篇经典论文，其中介绍了listing embedding和user/listing-type embedding两种召回算法。

listing embedding召回

listing就是Airbnb中的房源，所以基于listing embedding的召回，本质就是一个i2i召回。

Near

本来是想照搬word2vec，将用户的点击序列看成一个句子，认为一个滑窗内的两个相邻listing是相似的，它们的embedding应该接近。

但是仔细想想，以上想法存在严重问题

- word2vec建模的是语言模型的“共现性”，即哪些词语经常一起出现，因此需要一个滑动窗口限制距离
- 而这里，我们建模的是listing之间的相似性，难道只有相邻listing之间存在相似性？一个点击序列首尾的两个listing就不相似了吗？

所以，理想情况下，一个序列中任意两个listing，它们的embedding都应该是相近的。但是在实践中发现，这样的组合太多，所以Airbnb还是退回到word2vec的老路，即还是只拿一个滑窗内的中心listing与邻居listing组成正样本对。但是由于“最终成功预订”的那个listing有最强的业务信号，所以我们拿它与点击序列中的每个listing组成正样本对。这也就是Airbnb论文中“增加final booked listing作为global context加入每个滑窗”的原因。

Far

绝大部分负样本还是随机采样生成的。但是，Airbnb发现，用户点击序列中的listing多是同城的，导致正样本多是同城listing组成，而随机采样的负样本多是异地的，这其中存在的bias容易让模型只关注“地域”这个粗粒度特征。

为此，Airbnb在全局随机采样生成的负样本之外，还在与中心listing同城的listing中随机采样一部分listing作为hard negative，以促使模型能够关注除“地域”外的更多其他细节。

Embedding

特征只用了listing ID，模型也只不过是一个大的embedding矩阵而已。缺点是对于新listing，其id不在embedding矩阵中，无法获得embedding。

Pairwise-loss

使用sampled softmax loss。

user/listing-type embedding召回

Near

这个所谓的用user-type去召回listing-type，实际上就是u2i召回，只不过Airbnb觉得预订行为太稀疏，所以将相似的user聚类成user-type，把相似的listing聚类成listing-type。

既然如此，"Near"步骤与一般的u2i召回，别无二致。即，如果某user预订过某listing，那么该user所属的user-type，与该listing所属的listing-type就应该是相近的。

Far

绝大部分负样本还是随机采样生成的。除此之外，增加“被owner拒绝”作为hard negative，表达一种非常强烈的“user~item不匹配”。

Embedding

特征只有user-type ID和listing-type ID，模型也只不过拿ID当行号去embedding矩阵中抽取embedding。

Pairwise-loss

使用sampled softmax loss。

点评

Airbnb的两个算法，在Embedding和Pairwise-loss两个步骤上，都是标准操作，平淡无奇。

由于Airbnb论文的话术向word2vec“生搬硬套”，给人一种感觉，在学习listing embedding时增加final booked listing作为global context加入每个滑窗，在学习user/listing-type embedding时将user-type和listing-type组成异构序列，都是脑洞大开的创新。但从Near的角度来看

- word2vec建模的是词语间的“共现性”，listing embedding建模的是一个session中任意两个listing之间的“相似性”，而user/listing-type embedding建模的是user和listing之间的“匹配性”，三者的目标截然不同。
- **序列、滑窗只是word2vec刻画“共现性”所独有的概念，完全没必要出现在学习listing embedding和user/listing-type embedding的过程中。**
- 学习listing embedding时增加final booked listing作为global context，与其说是创新，不如说是考虑计算量之后的一种折中方案。
- 让user type和其预订过的listing type组成正样本对，从“匹配性”的角度来看，天经地义，完全没必要像论文中那样组成user+listing异构序列，从“共现性”的角度来解释。

在Far这个维度，Airbnb增加“同城负样本”和“owner拒绝负样本”，是根据业务逻辑增加hard negative的典型代表。

Youtube召回算法

Youtube在《Deep Neural Networks for YouTube Recommendations》一文中介绍的基于DNN的召回算法，非常经典，开DNN在召回中应用之先河，被业界模仿。

Near

u2i召回的典型思路：user与其观看过的视频，在向量空间中是相近的。

Far

论文中只提到了随机负采样，没有提到如何打压热门视频，也没有提到如何增加hard negative。

Embedding

- **user embedding**
  - 用户看过的视频的embedding，pooling成一个向量
  - 用户搜索的关键词的embedding，pooling成一个向量
  - 以上两个向量，加上一些用户的基本属性，拼接成一个大向量，喂入多层全连接(FC)进行充分交叉
  - 最后一层FC的输出就是user embedding
- **video embedding**
  - 特征只用了video id，模型也只不过是一个大的embedding矩阵
  - 整个模型在两个地方要用到video embedding，一个自然是最后计算user embedding与video embedding点积作为匹配分的时候要用到，另一处是用户看过的视频的embedding要参与生成user embedding
  - 这两处video embedding是否需要共享？原文中没有详细说明。我是偏向于共享的，一来降低模型规模，二来增加一些冷门video得到训练的机会。

Pairwise-loss

使用sampled softmax loss。

Facebook的EBR算法

Facebook在《Embedding-based Retrieval in Facebook Search》一文中介绍的算法。如果想了解更多的技术细节，请参考我的另一篇文章《负样本为王：评Facebook的向量化召回算法》

Near

u2i召回的典型思路：user与其观看过的视频，在向量空间中是相近的。

Far

绝大部分负样本依然是通过随机采样得到的。论文明确提出了，**召回算法不应该拿"曝光未点击"做负样本。**

另外，本文最大的贡献是提供了两种挑选hard negative的方案。Facebook的EBR与百度Mobius的作法非常相似，都是用**上一版本的召回模型筛选出"没那么相似"的<user,item>对，作为额外负样本，来增强训练下一版本召回模型。**具体做法上，又分online和offline两个版本

- **在线筛选**
  - 假如一个batch有n个正样本对，<$u^{(i)}, t_+^{(i)}$>，那么$u^{(i)}$的hard negative是利用上一轮迭代得到的召回模型，评估$u^{(i)}$与同一个batch的除$t_+^{(i)}$之外的所有t_+的匹配度，再选择一个与$u^{(i)}$最相似的作为hard negative。
  - 文章还提到，一个正样本最多配置2个这样的hard negative，配置多了反而会有负向效果。
  - 缺点是仅仅采用一个batch中的item作为hard negative的候选集，规模太小，可能还不足够hard。
- **离线筛选**
  - 拿当前的召回模型，为每个候选item生成item embedding，灌入FAISS
  - 拿当前的召回模型，为每个user生成user embedding，在FAISS中检索出top K条近邻item
  - 这top K条近邻item中，排名靠前的是positive，排名靠后的是easy negative，只有中间区域（Facebook的经验是101-500）的item可以作为hard negative。
  - 将hard negative与随机采样得到的easy negative混合。毕竟线上召回时，候选库里还是以easy negative为主，所以作者将比例维持在easy:hard=100:1
  - 拿增强后的负样本，训练下一版召回模型。

Embedding

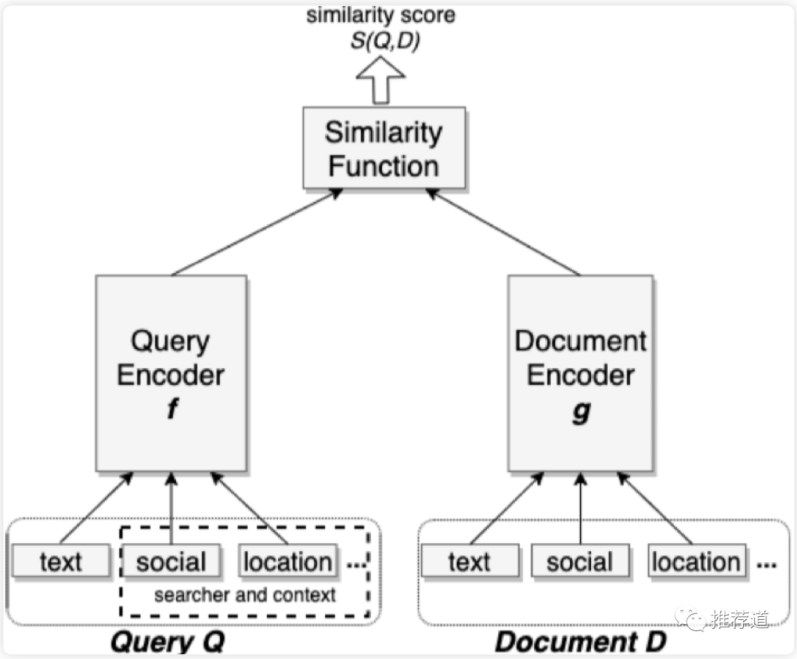
模型采用经典的双塔模型。**双塔模型鲜明体现了前文所述的"召回要求解耦"的特点**：

- 模型不会将user feature与item feature接入一个DNN，防止它们在底层就出现交叉
- user特征，通过user tower独立演进，生成user embedding
- item特征，通过item tower独立演进，生成item embedding

- 唯一一次user与item的交叉，只允许出现在最后拿user embedding与item embedding做点积计算匹配得分的时候

只有这样，才能允许我们

- 离线时，在user未知的情况下，只使用item tower，独立生成item embedding灌入faiss；
- 在线时，只使用user tower独立生成user embedding，避免与每个候选item进行“计算交叉特征”和“通过DNN”这样复杂耗时的操作



Pairwise-loss

使用margin hinge loss

Pinterest的PinSAGE

Pinterest 推出的基于 GCN 的召回算法 PinSAGE，被誉为"GCN 在工业级推荐系统上的首次成功运用"。对技术细节感兴趣的同学，推荐读我的另一篇文章《PinSAGE 召回模型及源码分析》。

Near

和Airbnb一样，我们可以认为被同一个user消费过的两个item是相似的，但是这样的排列组合太多了。

为此，PinSAGE采用随机游走的方式进行采样：在原始的user-item二部图上，以某个item作为起点，进行一次二步游走（item→user→item），首尾两端的item构成一条边。将以上二步游走反复进行多次，就构成了item-item同构图。

在这个新构建出来的item-item同构图上，每条边连接的两个item，因为被同一个user消费过，所以是相似的，构成了训练中的正样本。

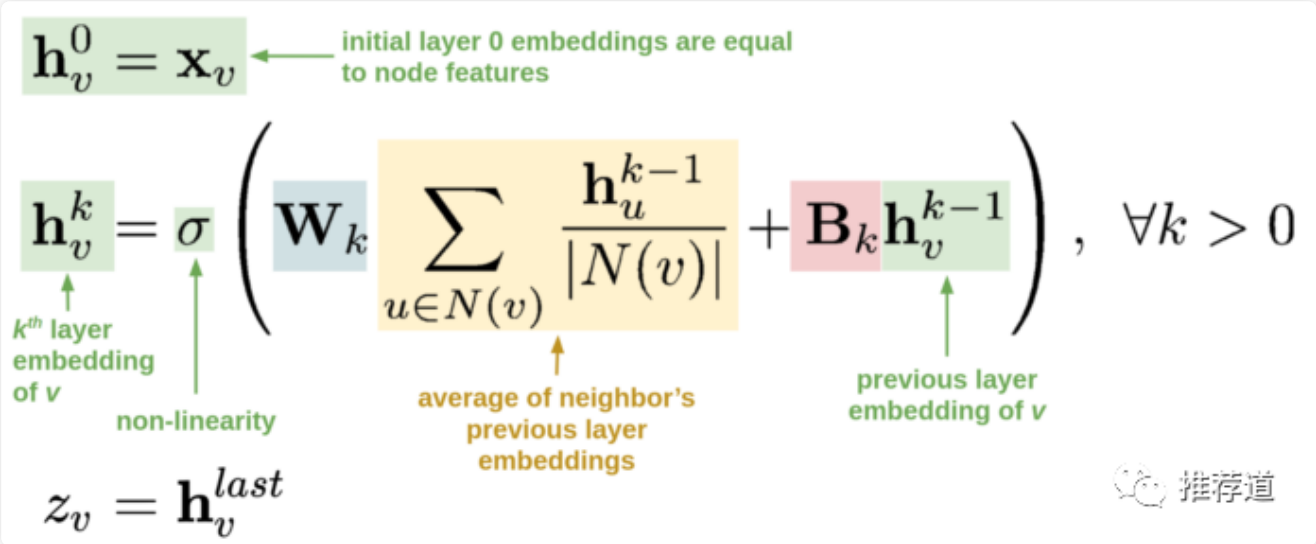
Far

PinSAGE提供了一种基于随机游走筛选hard negative的方法。

- 在训练开始前，
  - 从item-item图上的某个节点u，随机游走若干次。
  - 游走过程中遍历到的每个节点v，都被赋予一个分数L1-normalized visit count=该节点被访问到的次数 / 随机游走的总步数。
  - 这个分数，被视为节点v针对节点u的重要性，即所谓的Personal PageRank（PPR）。
- 训练过程中
  - 针对item-item同构图上的某一条边u→v，u和v就构成了一条正样本，它们的embedding应该相近
  - 在图上所有节点中随机采样一部分ne，u和每个ne就构成了一条负样本，它们的embedding应该比较远。因为是随机采样得到的，所以ne是easy negative。
  - 除此之外，还将u所有的邻居，按照它们对u的重要性（PPR）从大到小排序，筛选出排名居中（e.g. 论文中是2000~5000名）的那些item。**这些item与u有几分相似，但是相似性又没那么强**，从中再抽样一批item，作为"u"的hard negative。

Embedding

每个item embedding通过图卷积的方式生成。图卷积的核心思想就是：利用边的信息对节点信息进行聚合从而生成新的节点表示。多层图卷积的公式如下所示。



- 第一行，说明各节点h_v^0的初始表示，就等同于各节点自身的特征。这时还没有用上任何图的信息。
- 第二行，第 k 层卷积后，各节点的表示h_v^k,和两部分有关

- 第一部分，括号中蓝色+黄色部分，即先聚合当前节点的邻居的第 k-1 层卷积结果（$\sum_{u \in N(v)} \frac{h_u^{k-1}}{N(v)}$），再做线性变换。这时就利用上了图的信息，即某节点的邻居节点上的信息沿边传递到该节点并聚合（也就是卷积）
- 第二部分，括号中红色+绿色部分，即拿当前节点的第 k-1 层卷积结果（h_v^{k-1}），做线性变换
- 可以看到，**如果不考虑括号中的第一部分，这个公式简化为$h_v^k = \sigma(B_k h_v^{k-1})$，是不是很眼熟？这不就是传统的 MLP 公式吗？所以，图卷积的思想很简单，就是每层在做非线性变换之前，每个节点先聚合一次邻居的信息。**

- 第三行，最后一层卷积后的结果，成为各节点最终向量表示，用于计算两节点存在边的可能性。

Parwise-loss

使用margin hinge loss。

微信的GraphTR

这是我见过的最复杂的生成embedding的算法，没有之一。

Near

就是传统i2i召回的思路，在一个session内被观看的多个video之间是相似的，它们的embedding接近。

Far

就是随机负采样那一套，文中没有涉及打压热门视频，也没有涉及筛选hard negative。

Embedding

与上边介绍过的PinSAGE建立在item-item的同构图上有所不同，GraphTR建立在包括了user, video, tag, media (视频来源)这4类节点（每类称作一个域）的异构图上。每个节点要聚合来自多个领域的异构消息。

为了防止异构消息相互抵销而引入信息损失，GraphTR利用了三种聚合方式，**从三种不同粒度对于不同类型的邻居节点上的信息进行聚合**

- 将各域邻居节点的embedding拼接成一个大向量，按照传统的GraphSAGE方式聚合。这种聚合方式，不区分域，粒度最粗。
- 接下来的FM聚合，细致了一些，让不同域的邻居信息之间的两两交叉。
- 最后的Transformer聚合方式，粒度最细，不仅考虑了不同域之间的交叉，还考虑了一个域内部多个邻居节点之间的交叉。

为了生成item embedding，使用了GraphSAGE+FM+Transformer三种大杀器，是我见过的最复杂的向量化召回算法。对技术细节感兴趣的同学，可以参考我的另一篇文章《GraphSAGE+FM+Transformer强强联手：评微信的GraphTR模型》。

Pairwise-loss

sampled softmax loss

FM召回

与风头正劲的众多基于DNN/GNN的召回算法相比，FM召回算法，如今不太引人注目。但是，**FM召回性能优异，便于上线和解释，而且对冷启动新用户或新物料都非常友好**，仍然不失为召回算法中的一把利器。

Near

u2i召回的典型思路：user与其消费过的item，在向量空间中是相近的。

Far

绝大部分负样本还是随机采样生成的。同时在随机采样的过程中，要注意打压热门item，具体细节，见的我的知乎回答《推荐系统传统召回是怎么实现热门item的打压?》

至于如何增加hard negative来提高模型的精度，可以参考Facebook EBR中在线与离线筛选hard negative的做法。

Embedding

以上介绍的所有u2i召回（e.g, Youtube召回、Facebook EBR），召回的依据都只是user embedding与item embedding的点积，即**只考虑了user与item的匹配程度。**但是，**在一个实际的推荐系统中，用户喜欢的未必一定是与自身最匹配的，也包括一些自身性质极佳的item**（e.g.,热门视频、知名品牌的商品、著名作者的文章）。所以，我们在给某对儿<user,item>打分时，除了user/item的匹配度，**还需要考虑item本身的受欢迎程度。**

在FM召回中增加item自身得分非常简单，只需要将user embedding和item embedding都增广一维，如下图所示。其中E_{user}是某user包含的所有特征embedding之和，E_{item}是某Item包含的所有特征embedding之和。

$$E_{user}^{new} = concat(1, E_{user})$$

$$E_{item}^{new} = concat(\sum Item特征一阶权重 + \sum Item特征内部交叉, E_{item})$$

召回得分= $\langle E_{user}^{new}, E_{item}^{new} \rangle$

具体公式细节，请参考我的文章《FM：推荐算法中的瑞士军刀》中的FM召回一节。需要特别指出的是，**这种通过向量增广考虑“item本身的受欢迎程度”的做法，同样适用于其他u2i召回算法（e.g.,e.g, Youtube召回、Facebook EBR），有助于提高它们的精度。**

Pairwise-loss

在我的实现中，我使用了BRP loss。

总结

从NFEP框架视角来理解向量化召回算法，各算法的特点梳理如下表所示。

召回算法	模式	Near	Far	Embedding	Pairwise-loss
Airbnb listing召回	i2i	• 一个点击序列中的任意两个item都相近 • 为减少计算量，照搬word2vec，认为只在窗口内的两item才相近 • 将具备最强业务信号的item作为global context加入每个窗口	• 随机负采样 • 根据业务逻辑增加hard negative	只利用ID，从embedding矩阵中提取相应行	sampled softmax loss
Airbnb user/listing-type召回	u2i	user与其消费过的item，应该是相近的	• 随机负采样 • 根据业务逻辑增加hard negative	只利用ID，从embedding矩阵中提取相应行	sampled softmax loss
Youtube召回	u2i	user与其消费过的item，应该是相近的	随机负采样	• user embedding：将用户信息喂入DNN，DNN最后一层的输出 • item embedding：只利用ID从embedding矩阵提取相应行	sampled softmax loss
Facebook EBR	u2i	user与其消费过的item，应该是相近的	• 随机负采样 • 不应该用“曝光未点击”做负样本 • 用上一版模型筛选hard negative	双塔模型 • user特征，通过user tower独立演进，生成user embedding • item特征，通过item tower独立演进，生成item embedding	margin hinge loss
百度的孪生网络	u2u	将一个用户对item的交互历史拆解为两份， 其中一半历史与另一半应该是相似的	随机负采样	双塔模型 • 拿用户的一半交互历史喂入一塔 • 塔的最后一层输出，代表这一半交互历史的user embedding	bpr loss
Pinterest的PinSAGE	i2i	• 被同一个user消费过的item之间应该是相近的 • 为减少计算量，通过随机游走，对所有可能的item pair进行采样	• 随机负采样 • 通过Personal PageRank筛选hard negative	同构图上卷积， 通过聚合节点本身信息与邻居信息获得节点embedding	margin hinge loss
微信的GraphTR	i2i	在一个session内被消费的多个item之间是相近的	随机负采样	异构图上的卷积 • 既通过FM让不同领域的邻居之间相互交叉 • 又通过Transformer让同一领域的邻居之间相互交叉	sampled softmax loss
FM召回	u2i	user与其消费过的item，应该是相近的	• 随机负采样 • 打压热门item • 借鉴Facebook的方法增强hard negative	• user所包含的feature embedding相加得到user embedding • item所包含的feature embedding相加得到item embedding • 向量增广一维，以考虑item本身的受欢迎程度	bpr loss

主流向量化召回算法梳理

可以看到，通过NFEP框架的梳理，各召回算法间的异同变得清晰，便于我们加深理解，融会贯通。

而当你为自己的召回项目选型的时候，**你可能希望实现双塔模型，还希望模仿Facebook的做法来增强hard negative，同时还要像FM召回一样将“item本身受欢迎程度”考虑在内。**借助NFEP框架的梳理，我们可以更好地取长补短，设计出最适合你的业务场景、数据环境的召回算法。

- END -

图算法在网络黑产挖掘中的思考

DataFunTalk

入职半年小结 | 给应届校招算法同学的几点建议

蘑菇先生学习记