推荐系统召回策略—多路召回与Embedding召回

原创 stephen AI学习社区 7月25日

收录于话题 #推荐算法学习笔记 4个

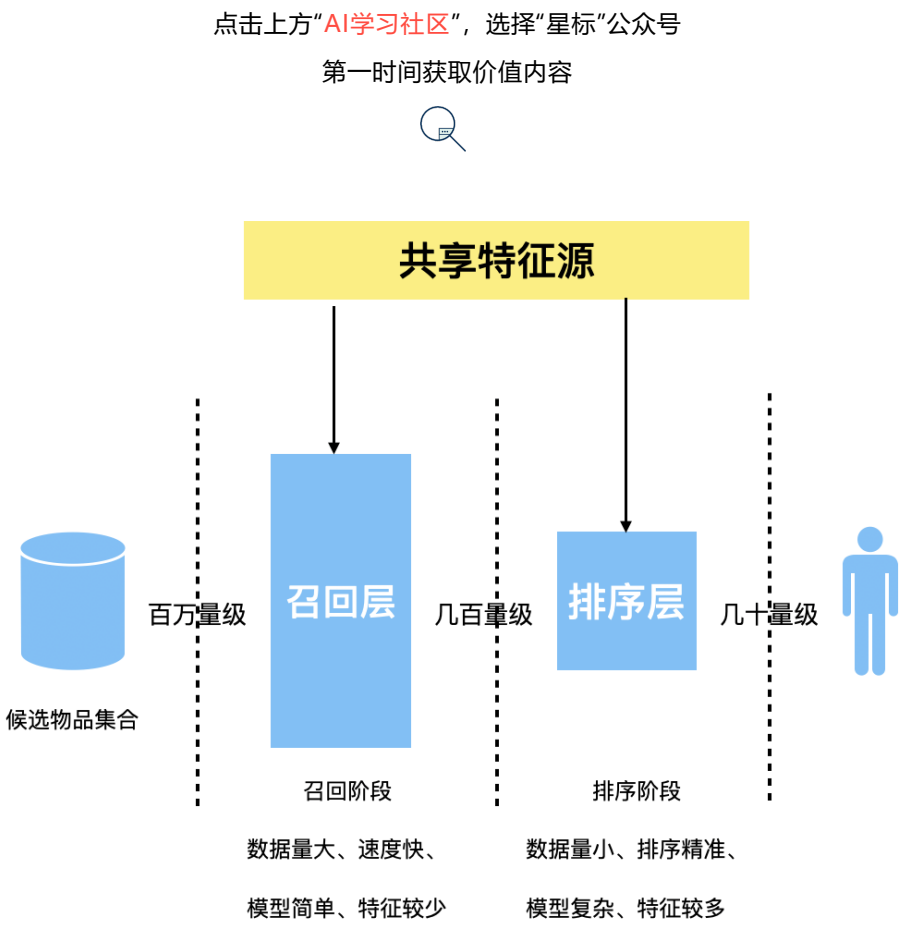

图1. 推荐系统整体架构

一. 多路召回

1.1. 概述

所谓的“多路召回策略”就是指采用不同的策略、特征或者简单模型，分别召回一部分候选集，然后再把这些候选集混合在一起后供后续排序模型使用的策略。

然后我们来说说为啥需要用到多路召回策略，我们在设计召回层的时候，“计算速度”与“召回率”这两个指标是相互矛盾的，也就是说在提高计算速度的时候需要尽量简化召回策略，这就会导致召回率不尽人意，同样的，需要提高召回率时就需要复杂的召回策略，这样计算速度肯定会相应的降低。在权衡两者后，目前工业界普遍采用多个简单的召回策略叠加的“多路召回策略”。

在多路召回中，每个策略之间毫不相关，所以一般可以写并发多线程同时进行。例如：新闻类的推荐系统中，我们可以按文章类别、作者、热度等分别进行召回，这样召回出来的结果更贴切实

际要求，同时我们可以开辟多个线程分别进行这些召回策略，这样可以更加高效。

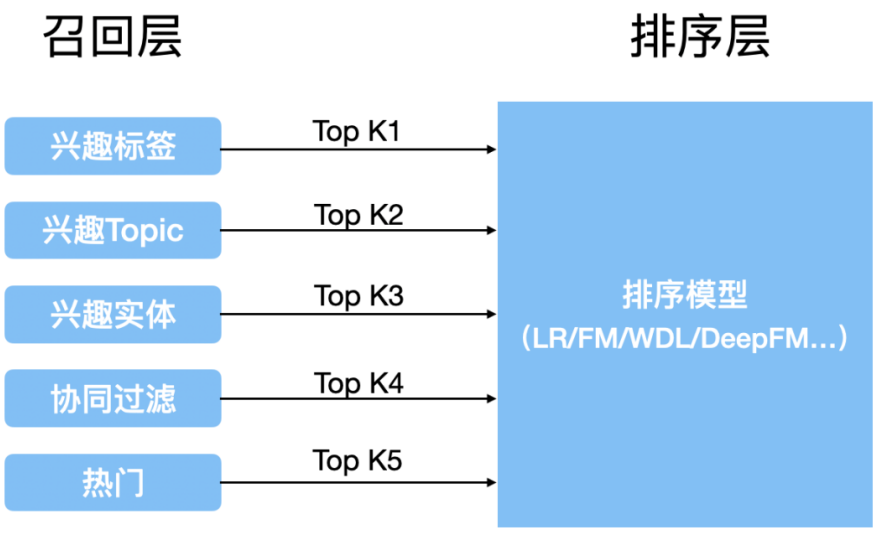

图2. 多路召回示意图

需要注意的是，在选择召回策略时需要充分考虑相关业务的特点，也就是说与业务强相关的。例如，对于新闻的召回来说，可以是“热点新闻”、“新闻类型”、“新闻类容”、“作者召回”等。

如上图2所示，每一路的召回都会拉取前K个候选集，对于每一路的K的大小属于超参数，可以不同。K的大小一般需要通过离线评估加上线上A/B测试的方式确定合理的取值范围。

虽然现在工业界普遍采用多路召回的策略，但是多路召回仍存在一些不可避免的缺陷，比如说，从策略选择到候选集大小参数的调整都需要人工进行，另外不同策略之间的信息也是割裂的，无法综合考虑不同策略对同一个物品的影响。当然，现在针对这些缺陷已经有了较好的解决方法——基于Embedding的召回，本文后面会讲到。

1.2. 多说一点

需要注意的是，在选择召回策略时需要充分考虑相关业务的特点，也就是说与业务强相关的。例如，对于新闻的召回来说，可以是“热点新闻”、“新闻类型”、“新闻类容”、“作者召回”等。

如上图2所示，每一路的召回都会拉取前K个候选集，对于每一路的K的大小属于超参数，可以不同。K的大小一般需要通过离线评估加上线上A/B测试的方式确定合理的取值范围。

虽然现在工业界普遍采用多路召回的策略，但是多路召回仍存在一些不可避免的缺陷，比如说，从策略选择到候选集大小参数的调整都需要人工进行，另外不同策略之间的信息也是割裂的，无法综合考虑不同策略对同一个物品的影响。当然，现在针对这些缺陷已经有了较好的解决方法——基于Embedding的召回，本文后面会讲到。

1.3. 融合排序与策略

在每个召回策略后都得到了一些候选集后，那么如何融合这些结果呢...

举个例子：几种召回策略返回的列表（Item-id，权重）分别为：

召回策略	召回item列表（Item-Id：权重）	召回item列表（Item-Id：权重）	召回item列表（Item-Id：权重）
召回策略1	A：0.9	B：0.8	C：0.7
召回策略2	B：0.6	C：0.5	D：0.4
召回策略3	C：0.3	D：0.2	E：0.1

针对上面我们有以下的融合策略：

1. 按顺序展示 比如召回策略“召回策略1”>“召回策略2”>“召回策略3”，那么就直接展示A、B、C、D、E即可
2. 平均法 在平均法中，分母为包含item列表中item-id的召回策略的个数，分子为权重之和。举例来说：1）.B的计算方法：（0.8 + 0.6)/2、 2）. C的计算方法：(0.7 + 0.5 + 0.3)/3
3. 加权平均 在加权平均法中，我们自己指定相关的权重。例如，我们给三种策略的权重指定为0.4、0.3、0.2，则B的权重为（0.4*0.8 + 0.6*0.3 + 0*0.2）/ （0.4+0.3+0.2）。这个方法有个问题就是，每个策略的权重是自己设置的，并不准确，所以，有动态加权法
4. 动态加权法 计算XYZ三种召回策略的CTR，作为每天更新的动态权重。那么如何计算每种策略的CTR呢，依然举例来说：

   展现日志-带召回源：X，Y，Z，X，Y，Z
   点击日志-带召回源：点击X
   则每种召回的CTR = 点击数/展现数（X:1/2）

这种方法的缺陷就是只考虑了点击率，并不全面。

5. 机器学习权重法 我们可以利用机器学习的方法来设置权重，比如说，使用逻辑回归LR分类模型预先离线算好各种召回的权重，然后做加权召回，这种方法考虑更多的特征以及环境因素，会更准确。

以上融合排序的方法，成本逐渐增大，效果依次变好，可以按照成本进行选择，或者可以使用A/B测试来确定使用哪种方法。

二. Embedding召回

2.1 概述

在上面提到了多路召回策略，也说到了它存在的不足之处，为了弥补多路召回的不足，就产生了现在的基于Embedding召回策略，另外，Embedding召回方法在实际效果以及速度上均不逊色于多路召回。事实上，多路召回中的“兴趣标签”、“兴趣Topic”、“热门”等信息均可以作为Embedding召

回方法中的附加信息融合进最终的Embedding向量中，换句话说就是结合了基于Embedding召回与多路召回两种方法。

我们在做Embedding召回时，可以把Embedding间的相似度作为唯一的判断标准，因此可以随意限定召回的候选集大小。

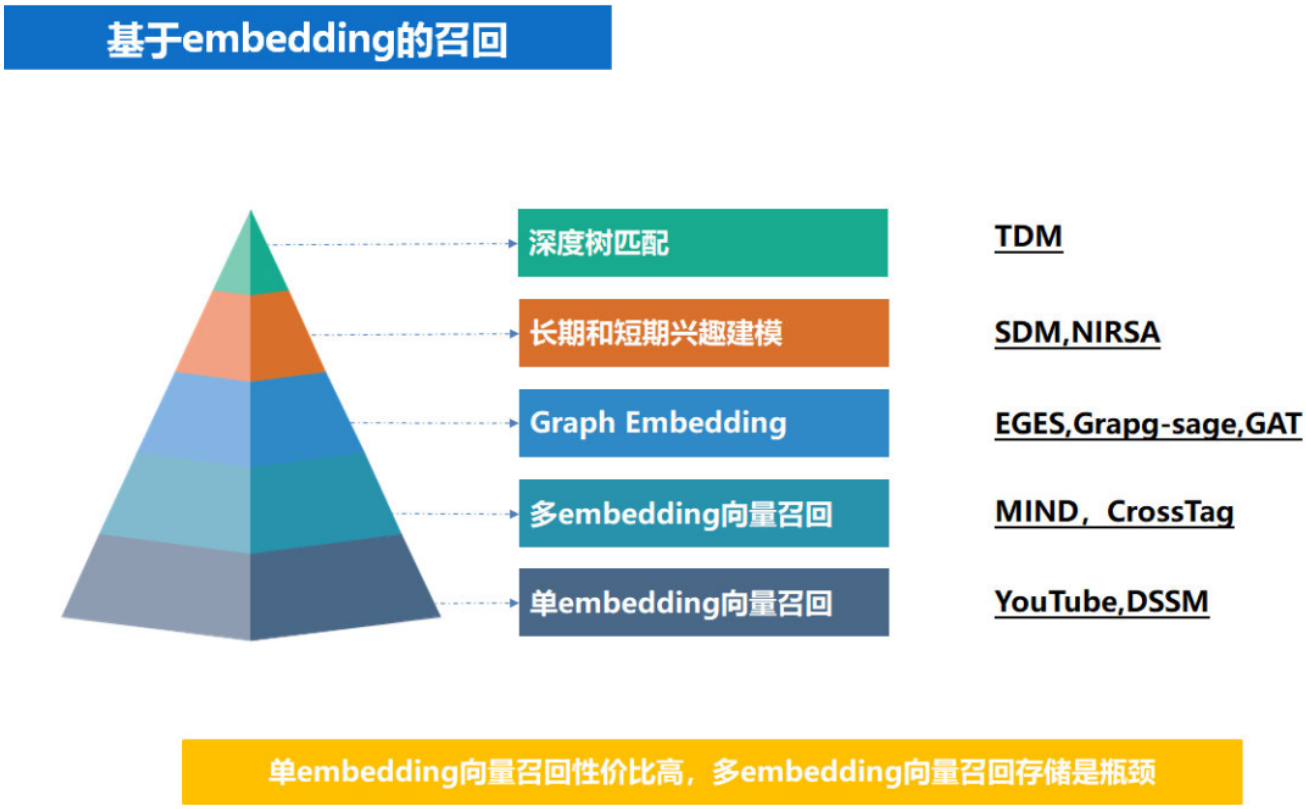

图3. Embedding召回方法总览

2.2 Embedding召回的常见方法

在介绍Embedding召回方法之前先简单介绍一些常见名词概念，如：I2I、U2I、U2U2I、U2I2I、U2TAG2I，如下图所示，其中“2”代表的是下图中边，“U”与“I”代表的事下图中的节点。

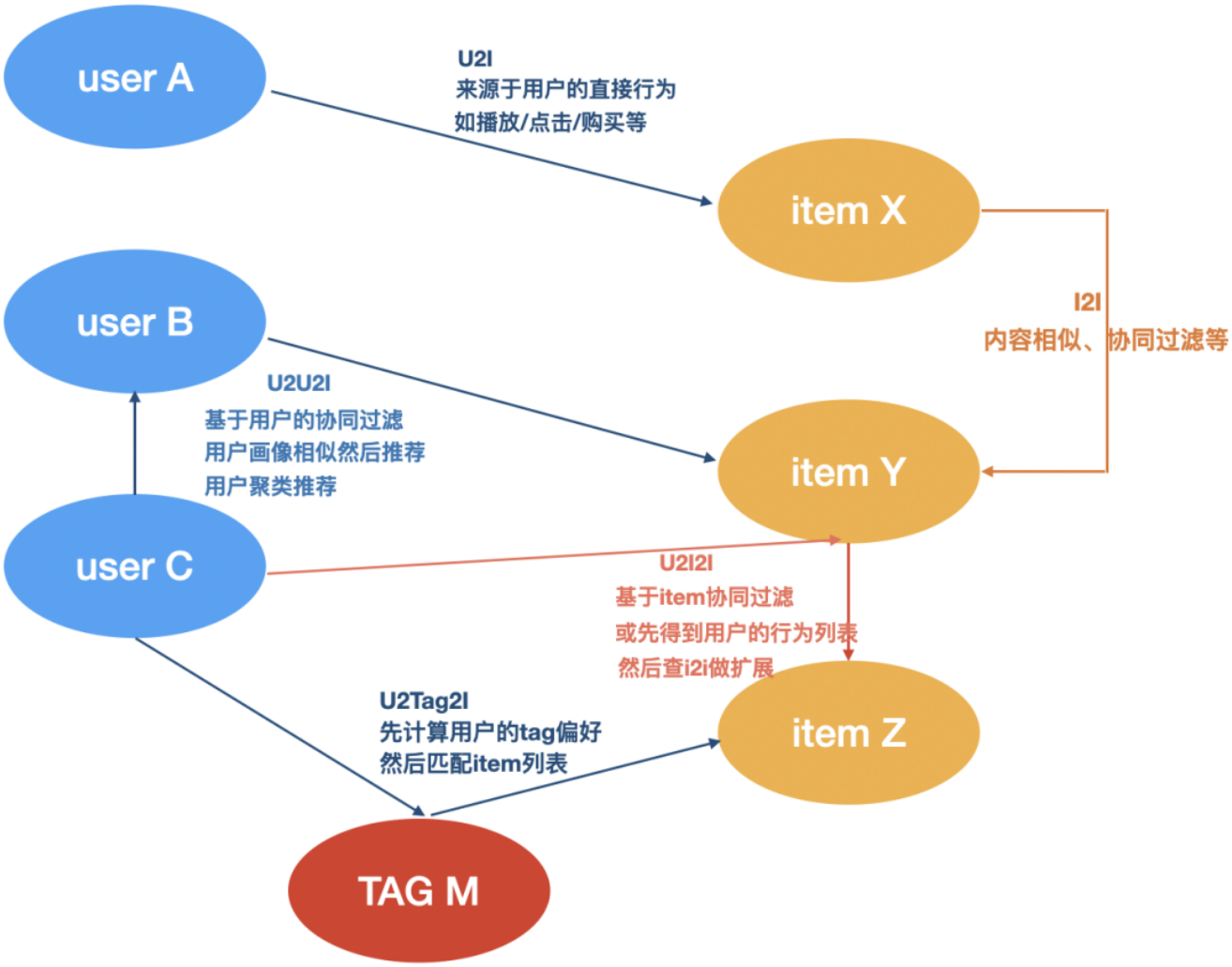

图4. 召回系统推荐路径

- I2I：计算item-item相似度，用于相似推荐、相关推荐、关联推荐；
- U2I：基于矩阵分解、协同过滤的结果、直接给u推荐i；
- U2U2I：基于用户的协同过滤，先找相似用户，再推荐相似用户喜欢的item；
- U2I2I：基于物品的协同过滤，先统计用户喜爱的物品，再推荐他喜欢的物品；
- U2TAG2I：基于标签的泛化推荐，先统计用户偏好的tag向量，然后匹配所有的item，这个tag一般是item的标签、分类、关键词等tag。

下问主要介绍的是最常用的U2I与I2I方法。

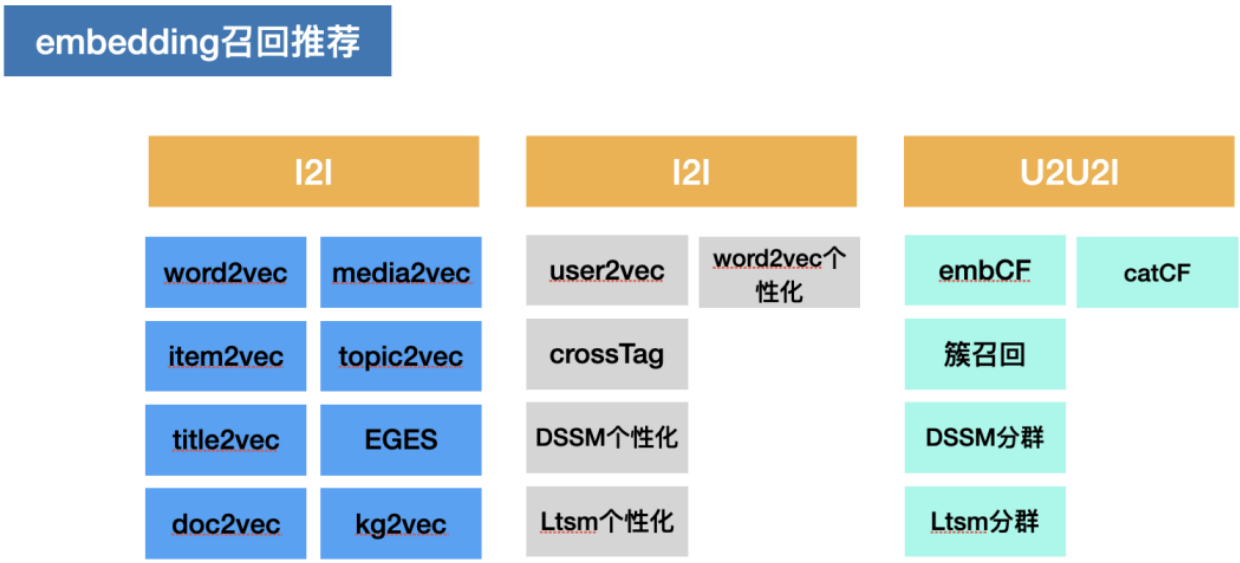

图5. embedding召回推荐常用方法

2.2.1 U2I召回方案

初步：u2i 召回算法实现了，uese2vec,word2vec 个性化,crosstag，DSSM 个性化等召回算法；user2vec 是拿用户的 tag 向量和文章的 tag 向量求相似度，做的召回；DSSM 个性化是拿用户的 DSSM 向量和文章的 DSSM 向量求相似度，做的召回；crosstag 相当于多个 user2vec,需要把用户的 tag 按类别进行统计，每个类取 K 个 tag,共获取 m 组 tag，然后各组分别做 user2vec,最后汇总得到用户的推荐列表。

进阶：uesr2vec 是在做召回的初级阶段，做的一些朴素的尝试，简单暴力见效快，存储压力大。每个 user 都存储一个推荐列表，在产品初期 DAU 不多时，矛盾还不明显，随着 DAU 不断提升，存储问题日益严重，这迫使我们想办法改变现状，可行的策略有两条，一个是把离线提前计算再存储转为线上即时计算不存储，另一个是把按人推荐转化为分群推荐。两种方法我们都做了实践。

分群召回流程大体如下：

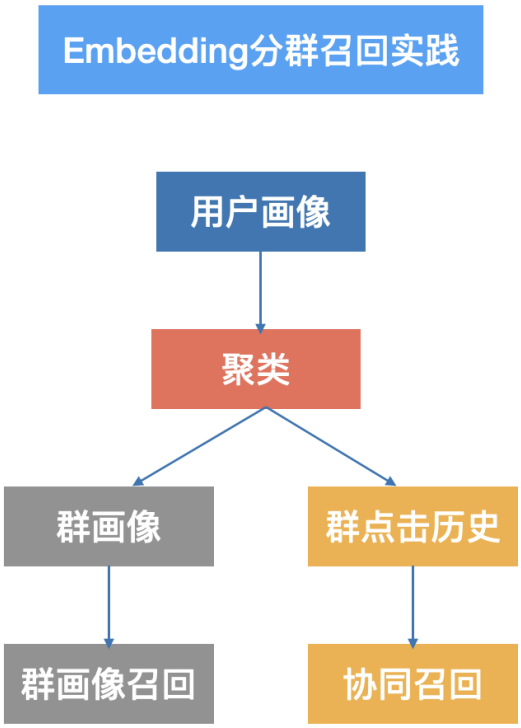

图6. Embedding分群召回流程

分群推荐我们尝试了簇召回，群画像召回，LSTM 分群，DSSM 分群，bnb 分群，增量聚类，动态规则聚类。

簇召回就是先把所有用户的 tag 向量用聚类算法（如 minibatch-kmeans）聚成若干个簇（比如 500 个，根据肘点法确定），然后保存下簇标签，簇中心，每个用户所属的簇（一个用户可以隶属于一个簇或者多个簇）。得到用户所在的簇后，有两种做法，一种是根据实时点击日志，在簇内做实时 CF，也就是在簇内把点击过的新闻相互推。另一种做法是离线定时计算各个簇中心和候选新闻的相似度，然后和到每个簇的候选集。从实验效果来看簇内做实时 CF 效果要好一些。

群画像召回是先把用户分群，然后把同一个群里的用户画像全部抽取出来，然后融合为一个群画像，相当于把这一群人合成了一个人，然后对于群画像，再使用和单个用户画像类似的个性化召回。

LSTM 分群和簇召回类似，不过用户的向量是通过用户最近点击文章的 m 篇文章的 bert 向量（tag2vec 向量亦可）送入 LSTM 得到用户的向量，剩下的步骤和簇召回类似，该算法有一定提升但是计算速度慢，很难铺量。

DSSM 分群，是把用户画像送入 DSSM，得到一个用户 64 维的向量，把文章画像送入 DSSM，得到一个文章的 64 维的向量，剩下的步骤和簇召回类似。该算法有提升显著，已经铺量使用。

bnb 分群是借鉴 airbn（爱彼迎）公布的房源推荐算法，把文章的多个特征的 embedding（tag，topic，cat）拼接成一个向量，类似得到文章的向量。剩下的步骤和簇召回类似，该算法有一定提升，不十分显著。

然后关于增量聚类与动态规则聚类后面有机会再补充说明吧。。。

2.2.2 I2I召回方案

单纯使用 fasttext+faiss 就可以实现好几路召回算法，比如：iten2vec,media2vec,tag2vec,loc2vec，title2vec。

tag2vec 就是利用词向量去做召回，比如可以用文章的标签向量表示文章的向量，如果一个文章有 4 个 tag（keywords: "蒋凡;离婚;张大奕;网红张大奕"）我们的经验是取前 3 个 tag,做等权重向量相加，效果最好。当然了这不是唯一的做法。关于 embedding 向量的用法有很多种比如，等权重相加，加权相加，取平均，取最大等。

得到文章向量之后就是典型的 item2item 的计算过程了，利用 faiss 计算每篇文章的相似文章，比如为每一篇文章查询询出 1000 篇候选文章后，按相似度作一个截断，比如 cosin sim<0.6 舍去，对余下的文章，再利用文章的其他特征比如热度，CTR，新鲜度作一个加权，一路最简单的 tag2vec 召回就诞生了。

其他召回和这个套路类似，就是训练 embedding 向量的时候，略有差异。tag2vec 是训练中文词语的向量，而 item2vec 是训练文章 ID（aid）所对应的向量，media2vec 训练的是文章的作者 ID（mid）所对应的向量，loc2vec 是训练地域名称所对应的向量，title2vec 是用 LSTM 训练得到的文章标题向量，doc2vec 是用 bert 计算出的文章正文（或者摘要）的向量。entity2vec 是利用我们自己构建的知识图谱通过 transE 得到的。

三. 自己的一些想法

推荐系统本身是与企业的业务强相关的，尤其是召回部分，所以说只要是合乎业务要求的召回策略均是合理的。所以本人感觉抛开业务谈召回其实有种“夸夸其谈”的感觉，当然写的这些召回策略肯定也是对实际工作有着一定的指导意义，可以借鉴这些思路再结合实际业务肯定能收获很不错的效果。

最后：因本人知识与视野的欠缺，本文有不足之处，诚恳接受批评！

参考：

1. 《深度学习推荐系统》 王喆著
2. 《推荐系统 embedding 技术实践总结》腾讯技术实践著

- END -

推荐系统学习笔记——特征工程
推荐系统召回策略—基于内容召回
推荐系统召回策略—基于协同过滤召回