12 | 基于热度的召回：如何使用热门内容来吸引用户?

2023-05-12 黄鸿波 来自北京

《手把手带你搭建推荐系统》

你好，我是黄鸿波。

在上一节课中，我们整体了解召回以及关于规则召回的相关知识，今天我们将聚焦在基于规则召回的另外一个分支：基于热度的召回。

基于热度的召回

基于热度的召回是召回算法中最直观且最简单的一种召回算法，它通过计算内容与用户之间交互的各个维度，从而设计出一套能够反映出内容受欢迎程度的算法，再根据这个算法进行召回。

我们通过微博这个 App 来简单说明一下基于热度的召回算法。

关注 推荐

热门 同城 榜单 明星 搞笑 情感

32 189 2205

不知名的孙小姐

搞笑幽默博主

+关注

这跟在大街上拉屎有什么区别？ 不知名的孙小姐的微博视频

搞笑幽默 迷惑行为 · 1.3万次观看

1 8 11

何凯文

3/10

刷博领现金

微博 发现 消息 我

我们可以看到在这个界面中的每一个条目中都有三个指标：转发、评论和点赞。当然，在画像中一定还有阅读指标。一般来讲，如果一篇内容的这四个指标越高，说明这篇内容就越受欢迎。

我们可以根据这四个指标的重要性，赋予相应的权重值。对于一篇文章来讲，如果没有任何阅读行为，我们可以认为这篇文章的热度值为 0，因为没有阅读行为也就不会有点赞，更不会有评论的产生。在这篇文章有阅读行为的基础上，如果这篇文章被用户喜欢，用户可能会点赞以及评论。如果非常喜欢，用户还会进行转发。

因此，对于转发、评论、点赞和阅读这四种操作类型，我们可以将其分别设置为权重为 0.4、0.3、0.2、0.1，针对上图中的帖子，假设这里的阅读数为 27000，点赞数为 11，评论数为 8，转发数为 1，就可以设计一个这样的公式。

$$C_{hot} = c_{count} \times 0.4 + d_{count} \times 0.3 + l_{count} \times 0.2 + r_{count} \times 0.1$$

在这里，我们使用 C_{hot} 代表内容的总热度值，c_{count} 代表转发数，d_{count} 代表评论数，l_{count} 代表点赞数，r_{count} 代表阅读数。我们用转发、评论、点赞、阅读的数值乘以其权重，最后再把所有的结果进行相加，从而得到最终的内容热度值结果。

$$C_{hot} = 1 \times 0.4 + 8 \times 0.3 + 11 \times 0.2 + 2700 \times 0.1$$

最终得到这篇内容的热度为 2705。我们通过这样的方式可以把每一篇内容的热度都计算出来，然后进行热度值的倒序排序，并取热度值的 topN 作为最终的召回集，从而进入推荐系统的下一个环节。

在这里，我们还要考虑一个特殊的情况：**如果短时间内一个关键词被频繁搜索或关注，就说明这个内容是一个最近突然非常热的话题，我们在求其热度值的时候就要做一个时间的加权。**

最常见的场景就是微博类的产品，当一个关键词被频繁的搜索或一个事件突然爆发，就会导致相关的内容被顶上热搜。此时我们就需要短时间内有内容的热度呈指数式上升，从而尽可能多地曝光给用户。

热度算法和热度的衰减

一个好的热度召回算法中的热度不仅只有增加，还需要有衰减的过程，而这个衰减一般是根据时间进行非线性衰减的过程。

首先我们要明确一下为什么要对热度进行衰减。我来举个例子：在春晚开始之前，大家都非常好奇春晚节目单，因此就会在搜索引擎上搜索与春晚节目单相关的内容。由于搜索、阅读、评论的量会非常大，因此在那段时间内的热度值就会非常高。但是当春晚过去了之后，大家对节目单的关注就会逐渐变小，因此也会减少对节目单的搜索。

但如果我们针对热度的设计仅仅是累加，就会导致与春晚节目单相关的内容依然排在前面，但这并不是用户想要的，所以我们需要在一定的时间内让热度降下去。总体来看，如果下降过慢会使热度持续时间过长，下降过快会导致内容被发现的时间较短，都不利用用户体验。因此，我们要设计一套热度衰减算法，使热度值的衰减能够在合理的时间预期内达到合理的热度值。

在热度衰减算法的设计中，我们可以借鉴牛顿冷却定律（Newton’s law of cooling）的思路来进行设计。牛顿冷却定律是指当物体表面与周围存在温度差时，单位时间从单位面积散失的热量与温度差成正比，比例系数称为热传递系数。在我们的算法设计中也可以引入冷却系数（或衰减系数），因此我们可以定义出下面这个算法。

$$T(t) = -\alpha(T(t) - H)$$

积分可得。

$$T = T_0 e^{-\alpha(t-t_0)}$$

其中 T_0 为上一期的热度，α 为冷却系数，t 为当前时刻，t_0 为初始时刻。由于指数函数受其指数幂 $(t - t_0)$ 线性函数的影响，导致衰减幅度过大。因此我们可以在原始函数的基础上，采用对数函数降低 $(t - t_0)$ 的影响程度，使曲线变得平滑，我们可以将上面的算法加以改造为下面这个。

$$T = T_0 e^{-\alpha log(t-t_0+1)}$$

化简后可得。

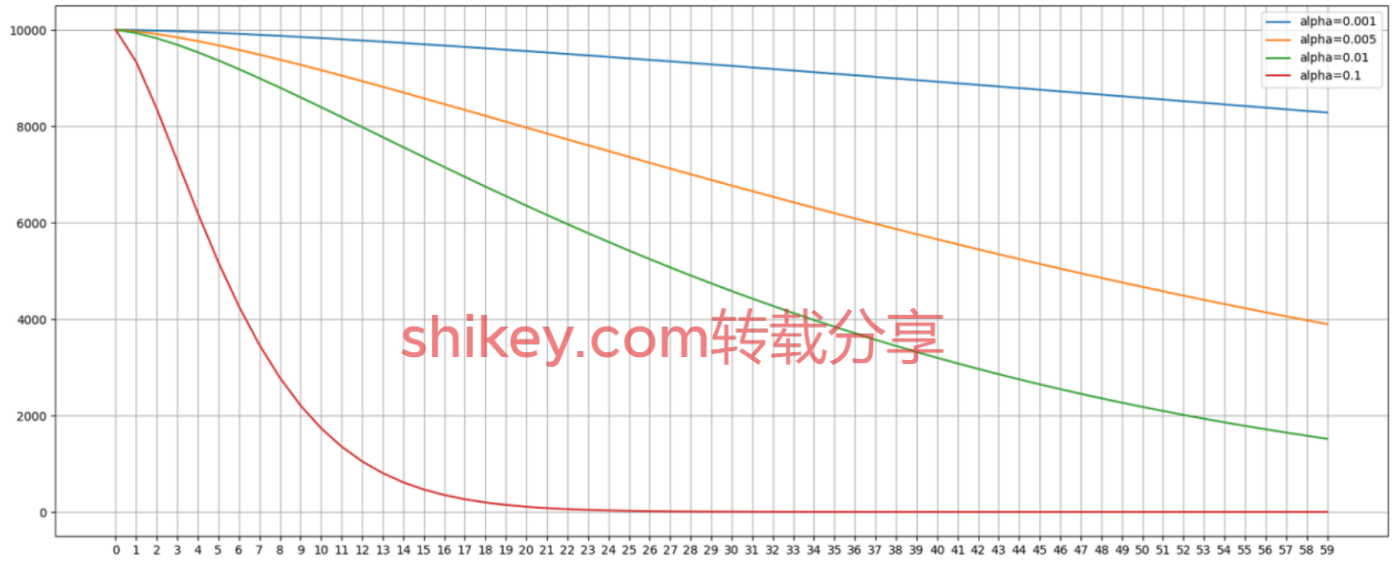
$$T = \frac{T_0}{(t-t_0+1)^{\alpha}}$$

我们可以使用代码来实现这个逻辑。

复制代码

```python
import matplotlib.pyplot as plt
import numpy as np
import math

def decay_function(alpha=0.01, init=10000, deltaT=100):
    data = []
    for t in range(deltaT):
        if len(data) == 0:
            temp = init /math.pow(t+1, alpha)#math.exp(-alpha * math.log(t + 1))
        else:
            temp = data[-1]/math.pow(t+1, alpha)# * math.exp(-alpha * math.log(t
        data.append(temp)

    plt.plot([t for t in range(deltaT)], data, label='alpha={}'.format(alpha))

init = 10000
deltaT = 60
plt.figure(figsize=(20, 8))

decay_function(0.001, init, deltaT)
decay_function(0.005, init, deltaT)
decay_function(0.01, init, deltaT)
decay_function(0.1, init, deltaT)

plt.xticks([t for t in range(deltaT)])
plt.grid()
plt.legend()
plt.show()
```

在上面的代码中，我们首先初始化了一个基本的热度值，在这里我们设置为 10000，每天的热度差值 deltaT 我们设置为 100。另外，我们取 4 种不同的热度衰减系数作为对比，运行代码后如图所示。



通过上面这张图我们可以看到，不同的热度衰减系数对同一个词的影响不同。在这里我们选取从热度衰减的第一天开始，为期 2 个月时间的热度值的变化。观察变化曲线我们可以看到，当热度衰减系数设置为 0.1 时（最下面那一条），在第 22 天热度值就几乎为 0；而当热度衰减系数设置为 0.001 时，到了第 60 天热度值仍然在 8000 以上。这两条线反映出热度衰减系数对热度值衰减过程影响程度的两个极端。

在实际工程项目中，你可以根据自身的需求和业务场景来制定热度衰减系数，并且对算法做一定的调整，从而做出符合自身业务的热度衰减算法。

基于热度的召回实现思路

在了解完热度召回和热度值衰减算法之后，接下来我们来看看基于热度的召回是如何实现的。

基于热度的召回算法从本质上可以分成两个部分，第一部分是召回算法和结果的实现，第二部分是热度值的更新。

我们可以简单理解为召回算法和召回结果的实现就是对热度值排序。在召回阶段中，为了呈现更好的推荐效果，我们还可以将其按照模块进行细分，然后再进行模块之间的内容组合。

假设在我们的 App 中有财经、体育、娱乐、教育这 4 个模块，每个模块下都有几千篇文章，现在要从这四个中的每个模块下选取若干文章进行推荐，这种召回方式我们叫基于模块的召

回。一般来讲，基于热度召回算法的实现都会与基于模块的召回进行配合，这样可以避免因为同一种类的内容热度过高导致内容同质化。

在实现过程中，我们需要对模块有一个基本的权重定义，我们可以根据用户的兴趣程度以及运营需求来设置模块的权重。例如在上面的例子中，我们可以假设财经、体育、娱乐、教育的权重分别为 0.4、0.3、0.2、0.1，假设目前我们每一个模块中都有上千篇文章，我们就可以从这四个模块中按照热度值倒序排序，然后分别取出相应条数的内容合成一个大集合，**这个大集合就是最终的召回结果。**

有了召回结果之后，接下来我们就要考虑召回结果的更新。因为每天都会有新的内容进来，每天也都会有新的热度值被更新，因此，我们的召回结果每天也都需要更新。在这里我们需要做一个定时任务，**每天通过定时任务对召回结果以及热度进行更新。**

对热度进行更新可以分成下面两种做法。

对热度进行累加更新。累加更新是指当一篇文章被进行阅读、点赞、评论、收藏等操作之后，会按照一定的比例数值将热度进行累加。

对热度进行衰减更新。衰减更新是指按照小时或者天的单位，对每一篇文章的热度进行热度衰减，这里具体的单位应该根据业务场景以及日均点击量来综合考虑得出。

接下来，我们根据数据集的内容对热度进行一个简单处理。

针对于我们的数据集，我们可以将每一篇文章都设置初始热度值为 1000。由于每篇文章都会有相应的行为信息，我们可以根据行为信息对热度进行累加。在我们的这个数据集中，行为信息主要包括 3 个：点赞、收藏和阅读。还记得我们在内容画像中有这几个关键的指标吗？实际上，在我们的项目运行时，这几个指标就会产生作用，从而根据一定的规则，将它们换算成热度值，然后当推荐的时候，我们就可以基于热度来进行推荐。

总结

我们来回顾一下本节课的主要知识点。

1. 基于热度的召回是召回算法中最直观也是最简单的一种召回算法。

2. 你应该对牛顿冷却定律有一定的了解，这个在我们的作业中也会有涉及。

3. 你需要知道基于热度召回的热度值是怎么算出来的，并且熟悉怎么做热度值的变化更新。

4. 了解基于热度召回的实现思路。

课后题

最后给你留 2 个小作业。

1. 实现牛顿冷却定律算法。
2. 试着将牛顿冷却定律运用到我们的项目中。

期待你在留言区与我交流讨论，我们下节课再见。

精选留言 (2)

peter

2023-05-16 来自北京

请教老师几个问题：
Q1：权重的设置有什么根据？完全是经验值吗？还是说有一个计算公式？本专栏类似于培训课程，在公司的实际开发中也是凭经验设置权重吗？
Q2：衰减系数公式的设置，也不是理性推导得来的，似乎也是根据经验，理论性不强啊。公司实际开发中也是这么实现的吗？
Q3：权重设置、衰减系数计算，是否用到了仿真？

> 作者回复: A1：这里的权重一般是由产品经理来评估每个参数和指标对于整个产品的重要性，以此来进行设置，没有具体的公式，在实际工作中也是如此，都是凭借经验然后慢慢调整；

A2：衰减实际上是根据牛顿冷却定律的思路来做的，在实际工作中一般也是沿用这个思路来做；

A3：没太理解这里的仿真指的是什么？

Geek_ccc0fd

2023-05-12 来自广东

实现代码：

```
from dao.mongo_db import MongoDB
import datetime
import math
class ContentLabel(object):
    def __init__(self):
        self.mongo_recommendation = MongoDB(db='recommendation')
        self.content_label_collection = self.mongo_recommendation.db_recommendation['content_label']

    def get_data_from_mongodb(self):
        datas = self.content_label_collection.find()
        return datas

    def update_content_hot(self):
        datas = self.get_data_from_mongodb()
        for data in datas:
            self.content_label_collection.update_one({"_id": data['_id']}, {"$set": {"hot_heat": self.hot_time_alpha(data['hot_heat'], data['news_date'])}})

    def hot_time_alpha(self, hot_value, news_date, alpha=0.01):
        # 计算当前时间和新闻时间天数差
        day = (datetime.datetime.now() - news_date).days
        hot = hot_value / math.pow(day, alpha)
        return hot
```

作者回复: 感谢你提供的代码