

15 | 基于Flask的推荐服务：如何把召回集推荐出去？

2023-05-19 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。

在前面的课程中，我们搭建了一个简单的 Flask 服务，并且已经可以通过 Postman 来进行调用，这节课我们将在此基础上，把基于规则的召回集成进来并推荐给用户。这节课你会学到下面的内容。

1. 写一个基于时间的召回，并存储到 Redis 数据库中。
2. 编写一个翻页查询服务，能够进行翻页查询。
3. 编写 Service 服务，将基于时间的召回推荐给用户。

编写基于时间的召回

推荐系统如果想要将内容推荐给用户，首先要做的就是找到合适的内容，然后将这些内容通过一定的整理和排序，按照一定的规则推荐给用户。这些规则可能是时间、热度、相似度以及

一些用于特征的评分，也可以是这些中一个或者多个算法的结合。在这节课里，我们就先用简单的基于时间的召回算法来做。

首先，我们来回顾一下我们之前的内容画像，下图是内容画像其中的一条数据。

```
1 17
{
  "_id": ObjectId("640e0cbd8b8b0c32ee9234bc"),
  "describe": "%黑龙江政务"微信公众号消息, 3月5日下午, 出席十四届全国人大一次会议的黑龙江代表团召开第三次全体会议, 审议政府工作报告, 中共中央政治局委员、中央政法委书记陈文清同代表们一起审议, 国务院机关党组副书记吴政隆参加审议
  "type": "news",
  "title": "江苏省委原书记吴政隆已在国务院机关党组副书记",
  "news_date": ISODate("2023-03-06T08:02:00.000Z"),
  "words_num": 1334,
  "hot_heat": 10000,
  "likes": 0,
  "read": 0,
  "collections": 0,
  "create_time": ISODate("2023-03-12T17:32:45.527Z")
}
```

我们可以看到，我们的用户画像实际上是由 10 个字段组成，在这 10 个字段中，第一个字段 `_id` 是 MongoDB 数据库为我们生成的一个唯一的 id 值，我们可以用其作为索引，来标记其唯一性。

与时间相关的字段有 2 个，一个是 `news_date`，另一个是 `create_time`。在这两个字段中，`news_date` 表示的是新闻发布的时间，`create_time` 指的是这个新闻的入库时间（也就是爬虫爬取的时间），这两个时间作为特征数据在不同的算法中有不同的用处。我们目前是想要基于时间来进行召回，这个时间最好使用新闻的发布时间，因为新闻发布时间属于新闻本身的一个特征，可以防止“时间穿越”事件的发生。

“时间穿越”是指在现在的时间点出现了未来的内容。比如说，我们在 2023 年 3 月 16 日爬取了一批 3 月 1 号之前的新闻，按理来说爬取的内容都是在 3 月 1 日之前的数据，这时突然出现了一个 3 月 15 号的新闻事件，虽然 3 月 15 日在现实生活中已经过去，但是在数据库里是一个未发生的数据，我们就称之为“时间穿越”。**在推荐系统中，要尤其注意避免“时间穿越”问题。**

我们再来想另外一个问题，既然我们要将内容推荐给用户，什么方法是最高效的呢？

如果我们直接从 MongoDB 中查询我们想要的数据库，然后再组装到用户界面上，效率就会变得非常低。因为首先 MongoDB 是文档型数据库，对于这种快速存取本身不是特别擅长，此外我们还要对 MongoDB 做按时间的倒序排序，本身也会有比较大的时间开销。所以在这里最好的解决办法就是使用 Redis 数据库来进行存取，这样能够使用户更加快速地得到内容。

总结一下，我们第一步是将我们的数据从 MongoDB 数据库中取出，然后按照时间的倒序进行排序存入到 Redis 数据库中。当需要给用户进行推荐的时候，我们直接从 Redis 数据库中读取数据，然后进行组装后推荐。

接下来我们来看看怎么实现这一步。

shikey.com转载分享

首先，在我们的 recommendation_class 项目中，有一个名为 scheduler 的目录。这个目录一般用来存放需要定时运行的任务，比如定时进行离线的召回、定时清除数据库中的无用数据、定时更新推荐列表等等。

随着时间的变更，爬虫新爬取出来的数据应该被及时加到里面，因此我们需要在里面新建一个 Python 文件，我们将其命名为 date_recall.py，然后在这个文件中做基于时间的召回。我们在做基于时间召回的时候需要设置一个时间范围，不需要对过于久远的内容进行召回。比如说超过一周的内容，我们认为就没有了时间的时效性。

还记得我们在讲爬虫的时候给你留的一个 [📧小作业](#) 吗？我们将数据重新爬取一遍并重新制作一下我们的内容画像，此时，你需要参考之前的课程完成以下 2 个步骤。


1. 将之前 MongoDB 数据库中 scrapy_data 数据库的 Collection 删掉，重新跑一遍爬虫程序，爬取更多的分类和更多的数据。
2. 将之前 MongoDB 数据库中 recommendation 数据库的 Collection 删掉，重新跑一遍内容画像中的数据。

我先来说一下我的新版代码（作业的拓展版）。在新版代码中我一共爬取了 3 个类别的数据，分别是国内新闻、电影、娱乐，并在 main.py 文件中设置了 1 个叫 page 的参数，目的是指定我爬取多少页的内容。这个时候，在爬虫页也会有对应爬取的页数设置。

经过 10 页数据的爬取，目前数据库总条数有 243 条，并将其制作成了内容画像。接下来，我们要将这一批数据按照时间顺序存入到数据库中。

由于我们所爬取的类别较少，并且娱乐和电影类别更新得又不是特别频繁，因此我们取国内新闻 20 天内的数据，娱乐和电影我们取 3 个月内的数据存入到 Redis 数据库中。

我们首先在项目中新建一个 Redis 数据库的连接工具类，用作连接 Redis 数据库。我们在 dao 目录下新建一个名为 redis_db.py 的 Python 文件，并编写如下代码。

 复制代码

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 import redis
4
5
6 class Redis(object):
7     def __init__(self):
8         self.redis = redis.StrictRedis(host= '127.0.0.1',
9                                         port= 6379,
10                                        password= '',
11                                        decode_responses=True)
```

这段代码非常简单，就是导入了 Redis 库，然后建立了一个数据库连接的变量连接到了本机，端口为 6379。

然后，我们再来编写 scheduler 目录下的 date_recall.py 文件。这个文件要从 MongoDB 中获取数据，然后按照时间的倒序插入到 Redis 数据库中。在这里，我们介绍一个 Redis 数据库中常用的数据结构 ZADD。

ZADD 命令用于将一个或多个成员元素及其分数值加入到有序集当中。如果某个成员已经是有序集的成员，更新这个成员的分数值，并通过重新插入这个成员元素来保证该成员在正确的位置上。分数值可以是整数值或双精度浮点数。如果有序集合 key 不存在，则创建一个空的有序集并执行 ZADD 操作。

在推荐系统中 ZADD 命令非常常用，我们可以将最外面的 key 用用户 id 或者推荐列表的类型（比如基于时间的召回、基于热度的召回）来代替，里面的 key 作为推荐内容的 content_id，value（也就是分数）作为相关度评分或者排序的顺序倒序。

当有用户需要推荐的时候，我们就从这里拿出相应的数据来进行推荐，比如一次取 10 个 content_id 作为推荐列表，我们来编写一下这个文件的代码。

```
1 from dao import redis_db
2 from dao.mongo_db import MongoDB
3
4
5 class DateRecList(object):
6     def __init__(self):
7         self._redis = redis_db.Redis()
8         self.mongo = MongoDB(db='Recommendation')
9         self.db_content = self.mongo.db_recommendation
10        self.collection_test = self.db_content['content_label']
11
12    def get_news_order_by_time(self):
13        ids = list()
14        data = self.collection_test.find().sort([{"$news_date", -1}])
15        count = 10000
16
17        for news in data:
18            self._redis.redis.zadd("rec_date_list", {str(news['_id']): count})
19            count -= 1
20            print(count)
21
22
23 if __name__ == '__main__':
24     date_rec = DateRecList()
25     date_rec.get_news_order_by_time()
```

这段代码也相对比较简单，首先我们从 MongoDB 数据库中读取数据，并按照新闻的时间进行降序排序，把它赋值给一个变量 data。然后我们再把数据插入到 Redis 数据库中，在插入到 Redis 数据库中的时候，我们使用的是 ZADD 方法。

ZADD 的优势是能够按照一定的顺序进行排序，这个顺序可以是升序也可以是降序。一般来讲，我们会以降序的方式排序，无论是时间还是分数都是如此。如果是时间降序的时候，就会将最新的内容排在前面，用户会看到比较新鲜的内容；如果是按照分数降序，就是将用户最感兴趣的内容放在前面，增加用户的点击率和用户的黏性。

在往 Redis 数据库中插入数据的时候，我们将 key 设置为 rec_date_list，这样的话所有使用时间召回算法的用户都会共用这一个列表，从而进行推荐。

除此之外，这里面我还设置了一个 count 值。在 ZADD 命令中以 score 作为排序依据，但是时间并不是一个 score。所以我在这里预先设置了一个比较大的值，然后按照这个值的降序给内容赋予分值，这样最高分代表了时间最靠前的内容，我们再进行推荐的时候，就可以按照这个方式进行推荐了。

运行上面的代码之后，我们在 Redis 数据库中查看一下结果。

ZSET: rec_date_list			重命名键
#	value	score	
1	642d978be51c641a885b09c4	9758	
2	642d978be51c641a885b09c3	9759	
3	642d978be51c641a885b09c2	9760	
4	642d978be51c641a885b09c1	9761	
5	642d978be51c641a885b09c0	9762	
6	642d978be51c641a885b09bf	9763	
7	642d978be51c641a885b09be	9764	
8	642d978be51c641a885b09bd	9765	
9	642d978be51c641a885b09bc	9766	

可以看到，我们的内容已经插入进去了。

到了现在，我们是不是就可以写 Service 的代码，然后将数据推送给用户了呢？当然可以，但是按照企业级的推荐方式，我不建议这么做。

首先，如果用这种方式我们需要从 Redis 中拿完每一个 id，然后再跑去 MongoDB 中查询相对应的内容，这样做实在是太慢了。用户量和内容量比较少的话还行，一旦我们的用户量以及数据库中的内容比较多，用这种方法就会导致推送的速度非常慢，从而影响用户体验。

比较好的一种方式就是将我们需要的内容存储到 Redis 数据库中，当列表需要查询数据时就直接从 Redis 中进行数据的调用，这样效率就会非常高。因此，我们在 scheduler 目录下新建一个 mongo_to_redis_content.py 文件，用来将 MongoDB 中的内容存到 Redis 数据库中，我们可以在里面编写如下代码。

复制代码

```
1 from dao import redis_db
2 from dao.mongo_db import MongoDB
3
4
5 class mongo_to_redis_content(object):
```

```

6     def __init__(self):
7         self._redis = redis_db.Redis()
8         self.mongo = MongoDB(db='recommendation')
9         self.db_recommendation = self.mongo.db_recommendation
10        self.collection_content = self.db_recommendation['content_label']
11
12
13    def get_from_mongoDB(self):
14        pipelines = [
15            {'$group': {
16                '_id': "$type"
17            }}
18        ]
19        types = self.collection_content.aggregate(pipelines)
20
21        for type in types:
22            collection = {"type": type['_id']}
23            data = self.collection_content.find(collection)
24            for x in data:
25                result = dict()
26                result['content_id'] = str(x['_id'])
27                result['describe'] = x['describe']
28                result['type'] = x['type']
29                result['news_date'] = x['news_date']
30                result['title'] = x['title']
31                self._redis.redis.set("news_detail:" + str(x['_id']), str(result))
32
33
34    if __name__ == '__main__':
35        write_to_redis = mongo_to_redis_content()
36        write_to_redis.get_from_mongoDB()

```

这段代码就是从 MongoDB 中获取内容，然后存储到 Redis 数据库中。只不过在这里，我们使用了一个新的 Redis 命令，叫做 SET。

Redis 的 Set 命令用于在 Redis 键中设置一些字符串值，在这里我们一个 key 可以对应一个字符串，也就是说我们可以将内容作为 value 传入进去。因此，我设置了 key 的值为 “news_detail:” 加上 content_id，这样就可以保证 key 的唯一性。然后再将 value 设置为内容，当有需要取内容的时候，我们就去取 “news_detail:content_id”。这样的话，整个效率会有质的提升。

我们运行上面的代码，可以得到下面的结果。



我们可以看到，现在 Redis 数据库中就是按照我们的设想来的，已经能够把每一个文章的详细信息都写入了进去。在这里，我们只需要内容的 id、内容本身、标题、类型和时间这 5 个字段即可。

将基于时间的召回推荐给用户

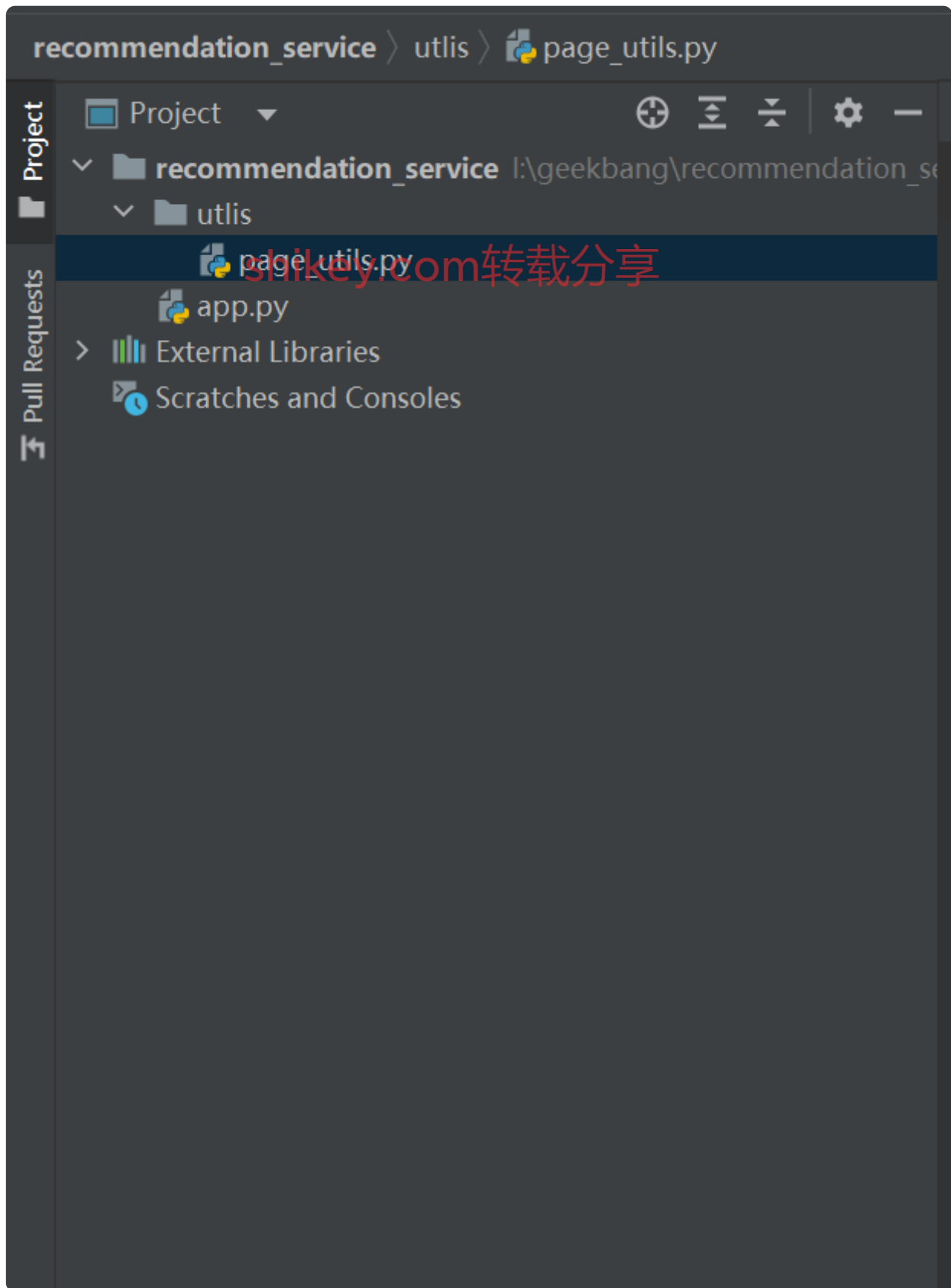
在前面的课程中，我们已经建立了一个最基本的 Flask 项目，接下来我们就要在这个 Flask 项目的基础上，开发我们第一个基于时间的推荐服务。

1. 一个简单的翻页功能，因为不可能每次都把所有的 id 传过去，所以加上一个翻页功能，能够让用户有更好的体验。
2. 将基于时间的召回集推荐给用户。

我们首先打开之前的 recommendation-service 这个项目，切换到我们的 recommendation-service 的 Anaconda 环境下，安装一些我们所需要的库，这里我们执行如下命令来安装 Redis 库。


```
1 pip install redis
```

然后在这个项目中新建一个叫 `utils` 的目录，主要是放置一些工具类。比如说我们的这个分页，实际上也可以看做是一个工具。然后我们在这个 `utils` 目录下，新建一个分页工具，名为 `page_utils.py`。然后我们新建一个 `dao` 目录，将我们的 Redis 数据库连接工具从其他项目中复制过来，命名为 `redis_db.py`，此时目录结构如下。



接下来我们来写这两个文件的代码，首先是 redis_db.py 文件的代码。

 复制代码

```
1 import redis
2
3
4 class Redis(object):
5     def __init__(self):
6         self.redis = redis.StrictRedis(host='localhost',
7                                         port=6379,
8                                         db=10,
9                                         password='',
10                                        decode_responses=True)
```

shickey.com 转载分享

上面这段代码我们前面多次用到，这里我们重点看一下 page_utils.py 文件的代码。

复制代码

```
1 from dao import redis_db
2
3 class page_utils(object):
4     def __init__(self):
5         self._redis = redis_db.Redis()
6
7     def get_data_with_page(self, page, page_size):
8         start = (page - 1) * page_size
9         end = start + page_size
10        data = self._redis.redis.zrevrange("rec_date_list", start, end)
11        lst = list()
12        for x in data:
13            info = self._redis.redis.get("news_detail:" + x)
14            lst.append(info)
15        return lst
16
17 if __name__ == '__main__':
18     page_size = page_utils()
19     print(page_size.get_data_with_page(1, 20))
```

这段代码主要是实现一个翻页的功能，我在里面实现了一个名字叫做 get_data_with_page() 的函数。在这里需要传入 page 和 page_size 两个参数，page 表示当前需要请求第几页，page_size 表示每一页有多少条内容。使用这种方法，对于前端界面的翻页和用户体验都有极大的帮助。

有了翻页功能之后，就可以正式写我们的接口程序了。接口程序在 app.py 这个文件里，原本的 app.py 是一个非常简单的程序，现在我们来对这个程序做一个改写。

```
1 from flask import Flask, request, jsonify
2 import json
3 from utlis.page_utils import page_utils
4 page_query = page_utils()
5
6 app = Flask(__name__)
7
8
9 @app.route('/')
10 def hello_world():
11     return 'Hello World!'
12
13 @app.route('/hello_rec', methods=["POST"])
14 def hello_recommendation():
15     try:
16         if request.method == 'POST':
17             req_json = request.get_data()
18             rec_obj = json.loads(req_json)
19             user_id = rec_obj["user_id"]
20             return jsonify({"code": 0, "msg": "请求成功", "data": "hello " + user_
21     except:
22         return jsonify({"code": 2000, "msg": "error"})
23
24 @app.route("/recommendation/get_rec_list", methods=['POST'])
25 def get_rec_list():
26     if request.method == 'POST':
27         req_json = request.get_data()
28         rec_obj = json.loads(req_json)
29         page_num = rec_obj['page_num']
30         page_size = rec_obj['page_size']
31
32     try:
33         data = page_query.get_data_with_page(page_num, page_size)
34         return jsonify({"code": 0, "msg": "请求成功", "data": data})
35     except Exception as e:
36         print(str(e))
37         return jsonify({"code": 2000, "msg": "error"})
38
39 if __name__ == '__main__':
40     app.run(port=10086)
```

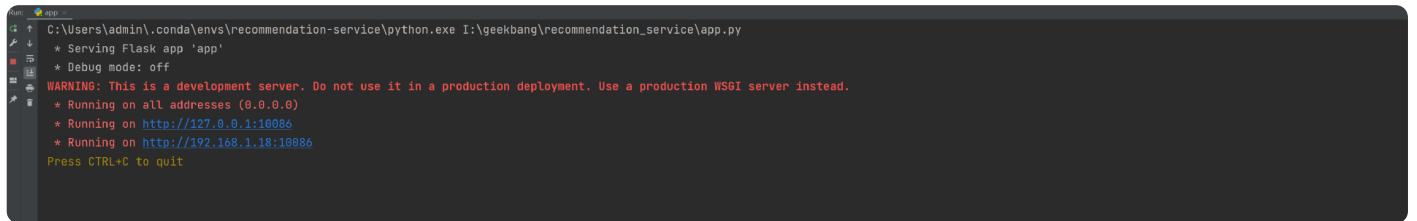
这段代码是一段 WebService 代码，就是把我们的服务提供成一个 API 接口进行调用，接收用户前端的请求。请求的参数有以下两个。

page_num, 表示当前请求的是第几页的内容。

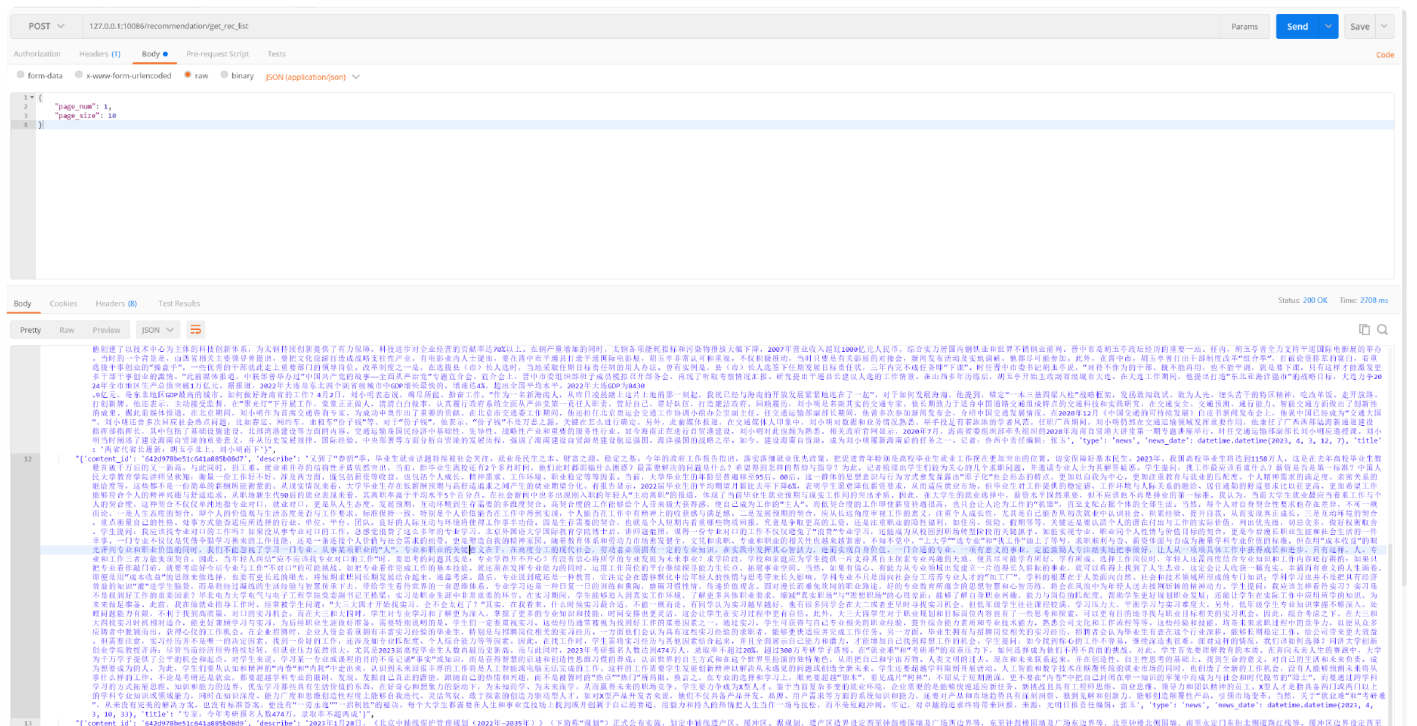
page_size, 表示当前请求的每一页有多少条数据。

当用户从客户端发送请求之后, 在 WebService 端就会使用 POST 接收这两参数, 然后传递到 page_query 的 get_data_with_page() 函数中, 这个时候, 函数内部就会根据请求的参数去数据库查询, 然后返回相应的结果。

下面, 我们来运行一下我们的 app.py 程序, 此时会得到如下结果。



现在我们的程序已经运行成功了, 接下来我们使用 Postman 来尝试调用一下它。



我们可以发现, 现在我们输出的内容就是按照时间倒序排序的, 并且, 我们调用时请求的参数为 page_num 和 page_size。

总结

现在我们已经能够把按照时间召回的内容推送出去了，今天我们主要讲了下面五个要点。

1. 基于时间召回可以和用户画像相结合进行召回。
2. 基于时间召回需要特别注意“时间穿越”的问题。
3. 你应该熟悉如何将 MongoDB 里面的数据按照时间顺序插入到 Redis。
4. 在推荐系统进行推荐的时候，知道如何做能够使推荐的效率更高、速度更快。
5. 我们可以使用翻页请求的方式来提高用户体验。

思考题

学完今天的课程，给你留两个小作业。

1. 复现今天的课程内容。
2. 给推荐过的内容存储在 Redis 已推荐列表中，并且下次推荐时候去除这一部分内容。

期待你的分享，如果今天的内容让你有所收获，也欢迎你推荐给有需要的朋友！

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (2)



peter

2023-05-20 来自北京

Q1: 源码链接已经提供了吗?

Q2: 百度首页会提供推荐列表，估计是什么算法?

Q3: 本文开始的“分数”，是根据什么确定的？某一个值可能选一个特殊值，比如特别大的数值，那其他的分数呢？

作者回复: A1: 同学您好，不好意思，最近事情有点多，源码将在下周左右提供；

A2: 一般来讲，都是用一個基本的推荐算法作为基础，然后配合深度学习的推荐来去做的，比如说Yo

uTubeDNN或者双塔模型，但是具体用的是是什么，这个我们无法判断；

A3：我们这里面的分数实际上是写死的一个序列，在用其他算法，比如说YouTubeDNN或者协同过滤的时候，会有具体的分数，那些是计算出来的。



Geek_ccc0fd

2023-05-19 来自广东

shikey.com转载分享

从mongodb获取排序数据报错： pymongo.errors.OperationFailure: FieldPath field names may not start with '\$'.

发现不需要带\$,我的pymongo版本4.3.3，正确代码：

```
data = self.collection_test.find().sort([{"$news_date", -1}])
```

作者回复：这个可能是因为版本的不同会导致细微的差异，大家遇到这种问题可以自行百度下，有好的实践经验，也可以一起分享下。

