

10 | 数据加工：如何将原始数据做成内容画像？

2023-05-08 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。

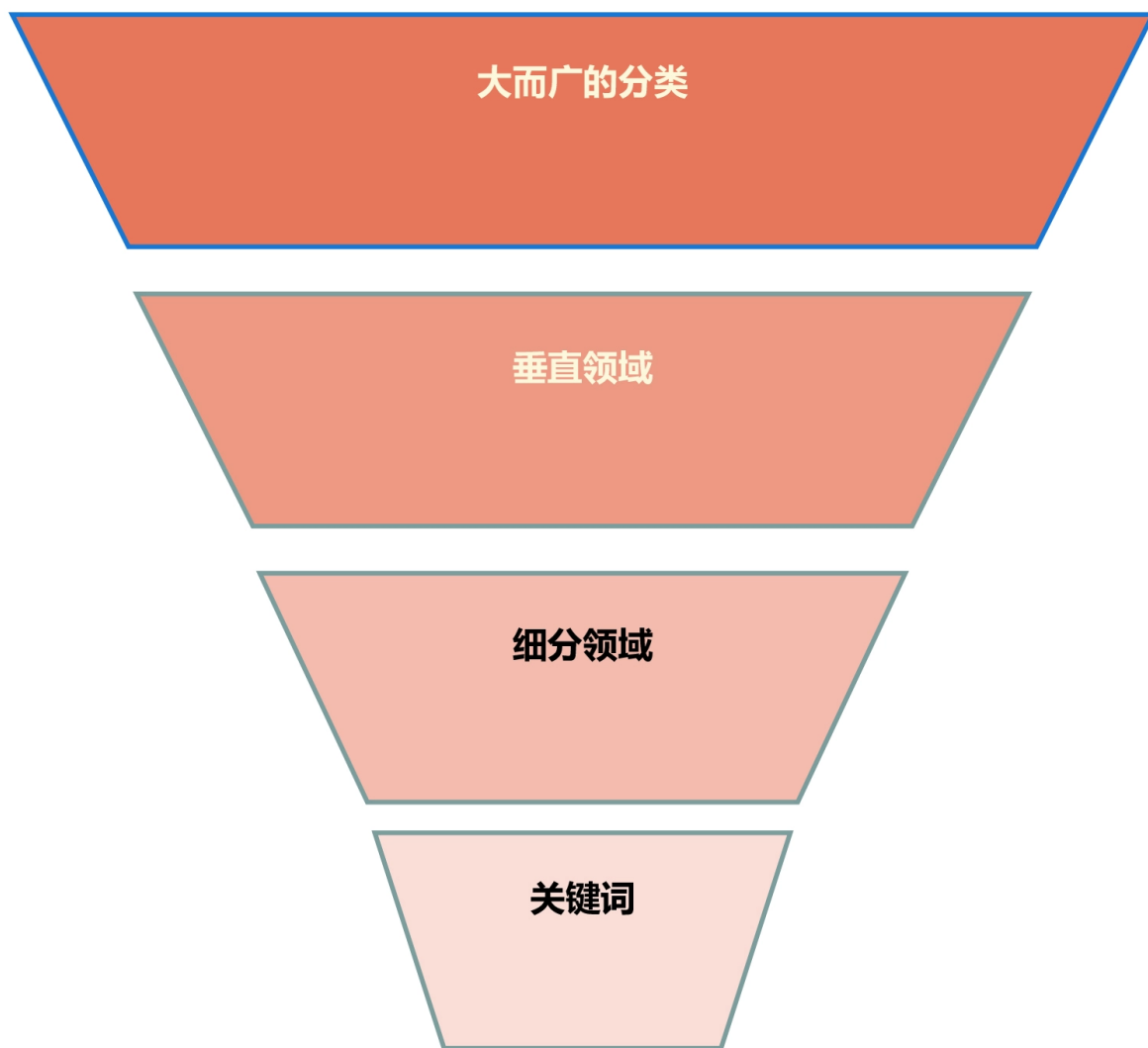
在前面的课程中，我们已经能够使用 scrapy 爬取想要的数据集，下面我们更进一步，把数据集处理成内容画像。这节课我会从内容画像的定义出发，带你了解内容画像的作用，紧接着，我们把原始的数据做成内容画像，直到最基础的画像已经能够正常写入到 MongoDB 数据库。

内容画像的定义与作用

从通俗的角度来说，内容画像实际上就是内容的一系列标签，我们在各个维度上给用户打上各种各样的标签，就组成了内容画像。由于内容在各个维度上被打上了不同的标签，因此，我们就可以在不同的维度上对内容进行分类。

内容的来源一般分成官方、用户和互联网（例如爬虫爬取），不同的来源肯定就会使得内容的形式、质量等都有比较大的区别。

从标签和分类的角度来讲，我们可以将内容标签呈现出漏斗式。也就是说，从一个大而广的分类到垂直领域，再到细分领域，最后到关键词这个级别。在这个漏斗中，每一个层级都可以作为画像中的一个标签或者一个特征，到实际的模型中再根据需求进行取出，从而进行模型的训练。



如果把内容画像平铺开，实际上我们得到的就应该是一个大的标签库。从这个标签库中随意抓出一个标签，就能找到这个标签所对应的内容的列表。当把标签进行各种组合时，就会产生

不同的列表。从理论上讲，组合的条件越多，所描述和刻画的标签也就越精细，所对应的内容也就更加具体，这对于判断用户的喜好来说是非常重要的。

内容画像是一个推荐系统推荐效果的核心所在，当我们在构建各种推荐算法和模型的时候，需要使用到各种各样的特征，而这些特征一般来讲都是从内容画像中获取到的各个标签，然后经过一系列的处理，得到我们需要的信息，从而进行推荐算法的模型构建和推理。

推荐系统包含内容画像和用户画像两个大的画像。实际上，用户画像也可以简单理解为内容画像中多个内容的集合。根据前面的讲解我们可以知道，用户画像里一般包含着一系列的用户行为，这些行为中很大一部分就是用户所浏览的内容信息。而一个用户之所以能够点击这些内容，或者能够对这些内容感兴趣，也都是因为这些内容的标签。

对于用户来讲，这些内容的标签可以是显性的也可以是隐性的。显性指很多产品明确标记了内容标签、关键词或者其他的信息；隐性指有些标签并没有单独以标签的形式标记出来，而是用户根据自己的判断得到的，比如说标题里面的关键词、摘要里面的关键短语等。实际上，这些标签都是真实存在的，而这些标签就组成了一个内容画像。

另外我们可以通过内容画像找到内容之间是否有共同标签，以及内容之间是否有一定的相似性。在实际推荐系统的运行过程中，就可以利用这些相似性给用户推荐相似的文章。

把原始的数据做成内容画像

下面我们来把原始数据做成内容画像。

我们可以将原始文本数据粗略分为**非结构化数据**和**结构化数据**。我们在处理不同类型数据时所用的方法略有区别，但最终想要达到的目的都是相同的，那就是提取出它们的标签。

非结构化数据是指数据结构不规则或者不完整，没有预定义的数据模型。目前大部分的原始数据都是非结构化数据，它广义上包括了文档、文本、图片、音频、视频等等。这节课主要指的是文本信息，更确切地来讲，就是从新浪网上爬取的内容数据。

这种非结构化文本一般包含了关键词提取、命名实体识别、文本分类、Word Embedding等，我给你画了一个表格，里面有这些技术对应的作用，你可以对照进行学习。

非结构化文本技术	作用
关键词提取	从文本和标题中提取标题或者文本中的关键词，可以使用如TF-IDF或者TextRank，也可以将两者结合使用
命名实体识别	可以使用命名实体识别技术来提取内容中的人名、地名、时间等常见信息，然后将其作为内容画像中的一部分
文本分类	当一篇新来的内容不知道具体应该放在哪个类别下时，可以将原有分类体系下的数据训练成一个分类模型，然后再拿新的数据进行预测，从而进行文本分类，这个分类的结果可以作为内容画像的一部分
Word Embedding	如果想要挖掘字面意义下的语义信息并进行维度的缩减，可以利用Word Embedding的形式进行画像特征的处理

极客时间

下面我们把之前使用 scrapy 爬取的数据制作成内容画像，然后存储到 MongoDB 数据库中。

首先我们拿 MongoDB 数据库中的任意一条数据为例。

```
1 //
2 {
3   "_id" : ObjectId("64074e12e3323cb2594ebc5d"),
4   "type" : "news",
5   "title" : "江苏省委原书记吴政隆已任国务院机关党组副书记",
6   "times" : ISODate("2023-03-06T08:02:00.000Z"),
7   "desc" : ""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"
8 }
```

在这条数据中，一共有 5 个字段，分别是 “_id”、“type”、“title”、“times” 和 “desc”。在这里我们可以发现，除了 “_id” 这个字段以外，其他的数据都是爬虫爬取回来的。这里 “_id” 这个字段，实际上就是 MongoDB 为这条数据建立的一个 id，并且这条 id 在 MongoDB 数据库中是唯一的。

想要把我们的内容做成画像，我会从下面几个角度来考虑。

1. 这个文章的标题包含了什么关键词？
2. 这个文章的内容包含了什么关键词？
3. 这个文章有多少个字，是长文还是短文？
4. 是否需要附上一个初始的热度？
5. 这个文章的类型是什么？

当然，我们所需要的不仅仅只有这些，不过前期可以先按照这几个角度来做成一个简单的内容画像。

一般来讲，内容画像不属于爬虫，它属于推荐系统的一部分。因此，在这里需要创建一个新的项目，用来专门做推荐系统与模型相关的处理，包括数据处理、模型搭建、训练等等。到了后面服务端内容的时候，我们也会再建立一个与服务端相关的项目。

这里需要注意的是，建立项目的时候，你需要选择一个 anaconda 的环境，一般来说，我对于 anaconda 环境的管理原则可以分成以下三个思路。


按项目。每建立一个项目就建立一个新的 anaconda 环境，并且环境名与项目名进行对应或者相关。

按环境相关性。所谓的按环境相关性就是指你要把你的常用的一类放在一个环境里，比如说某个环境就是和 TensorFlow 相关的库，或者某个环境就是和 pytorch 相关的库，不同的版本建立不同的库，这样的话可以做到环境的隔离，也相对比较通用。

按项目通用性。如果项目里面用不到太多太复杂的包，可以建立一个通用的 anaconda 环境，然后装一些通用的库（比如说 numpy、sklearn 等），一些简单的机器学习和科学计算的方法都可以使用这个包来做。

在这个项目里，因为要用到的环境相对比较复杂，所以我就新建立一个环境，将其命名为“recommendation-class”，使用的 Python 环境为 3.7，因此，我们在 cmd 环境下，输入下面这行代码。

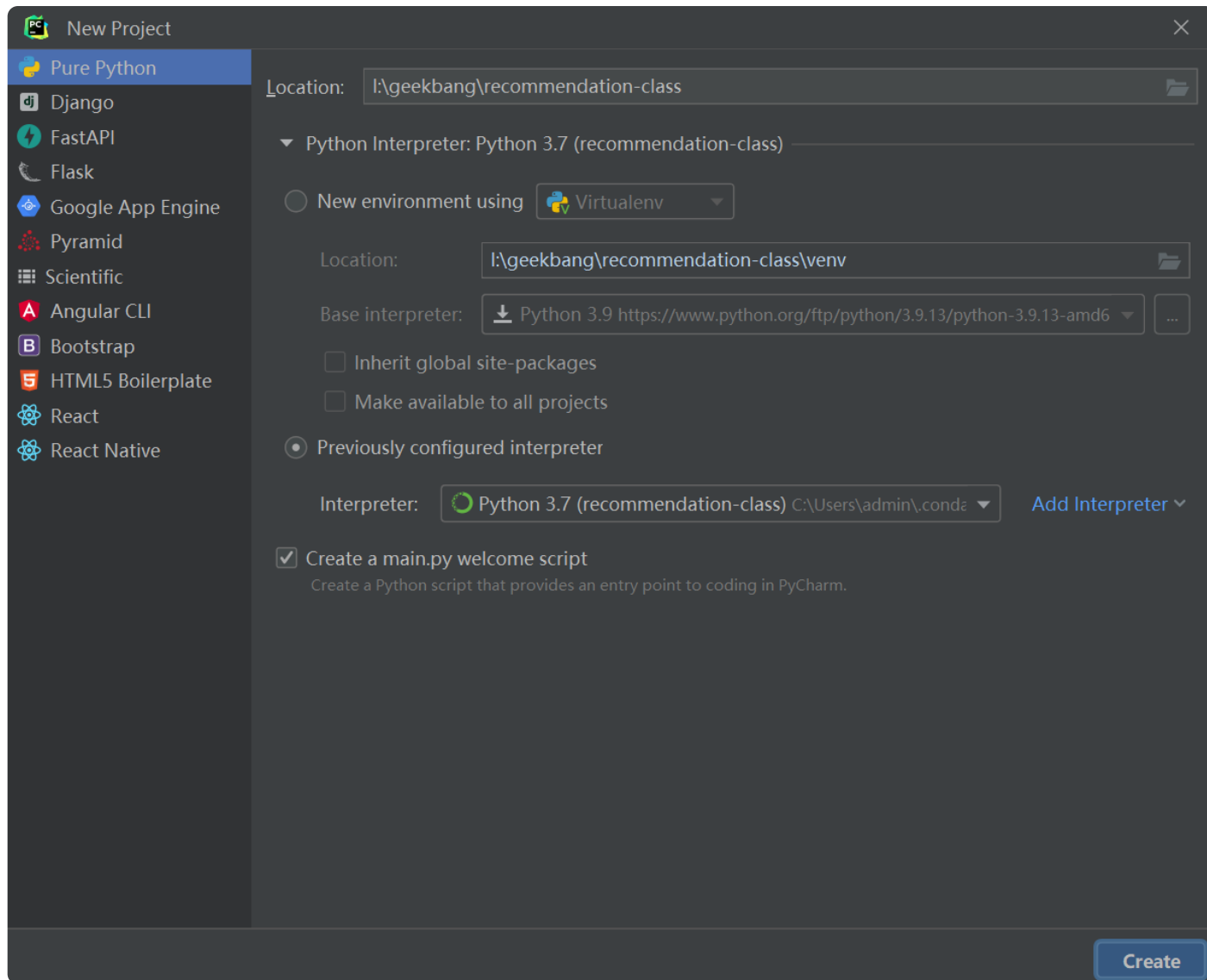
```
1 conda create -n recommendation-class python==3.7
```

 复制代码

创建完成后，使用下面的命令来进行激活环境。

```
1 activate recommendation-class
```

这个时候，暂时可以先不用安装需要的库，我建议先在 pycharm 中创建一个 recommendation-class 的项目，然后关联我们的环境，如图所示。

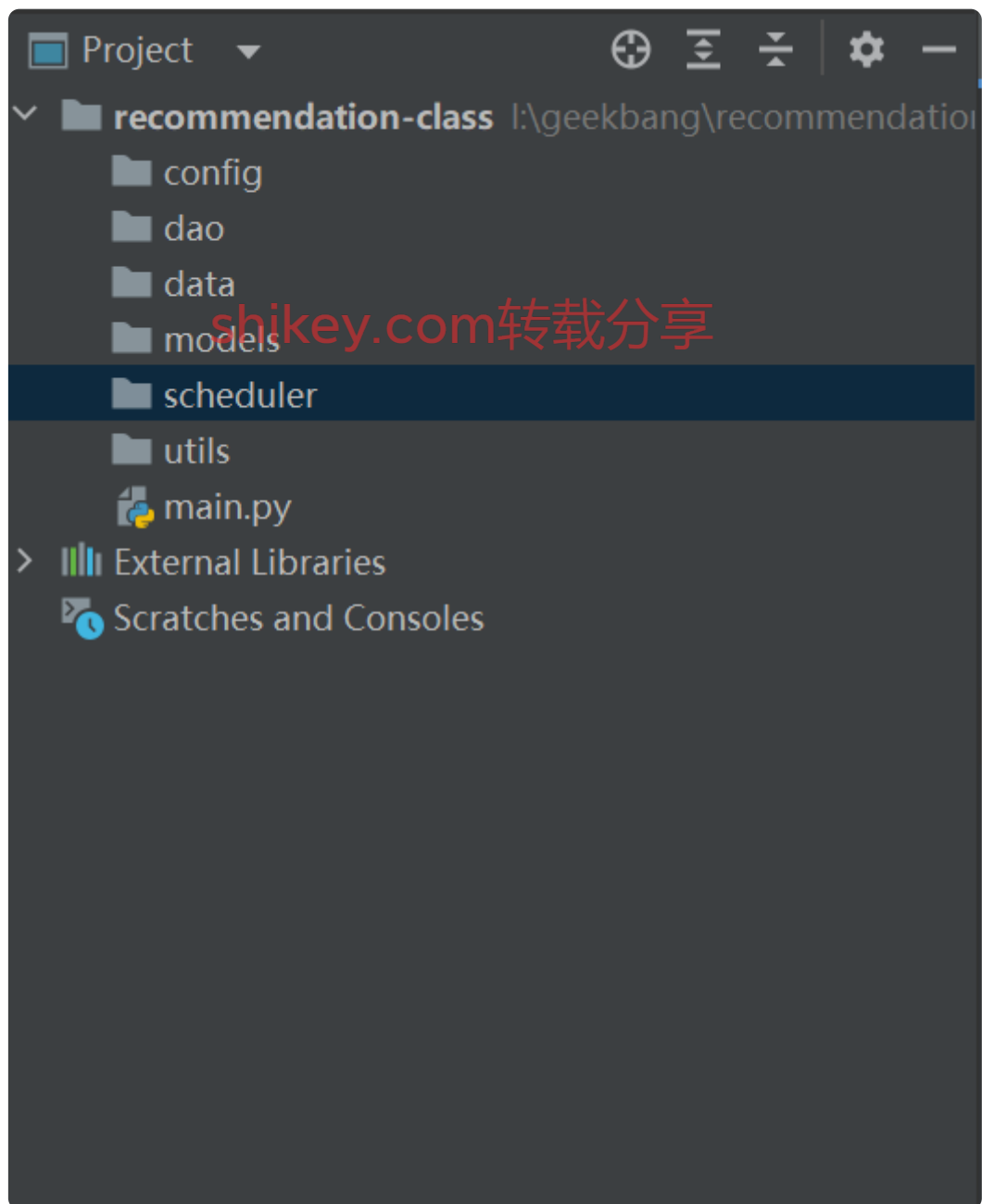


当项目创建好之后，就会出现下面的界面。



现在我们真正进入到推荐系统的代码开发部分。在开发代码之前来先对目录做一个规划，看看我们到底都需要哪些目录。

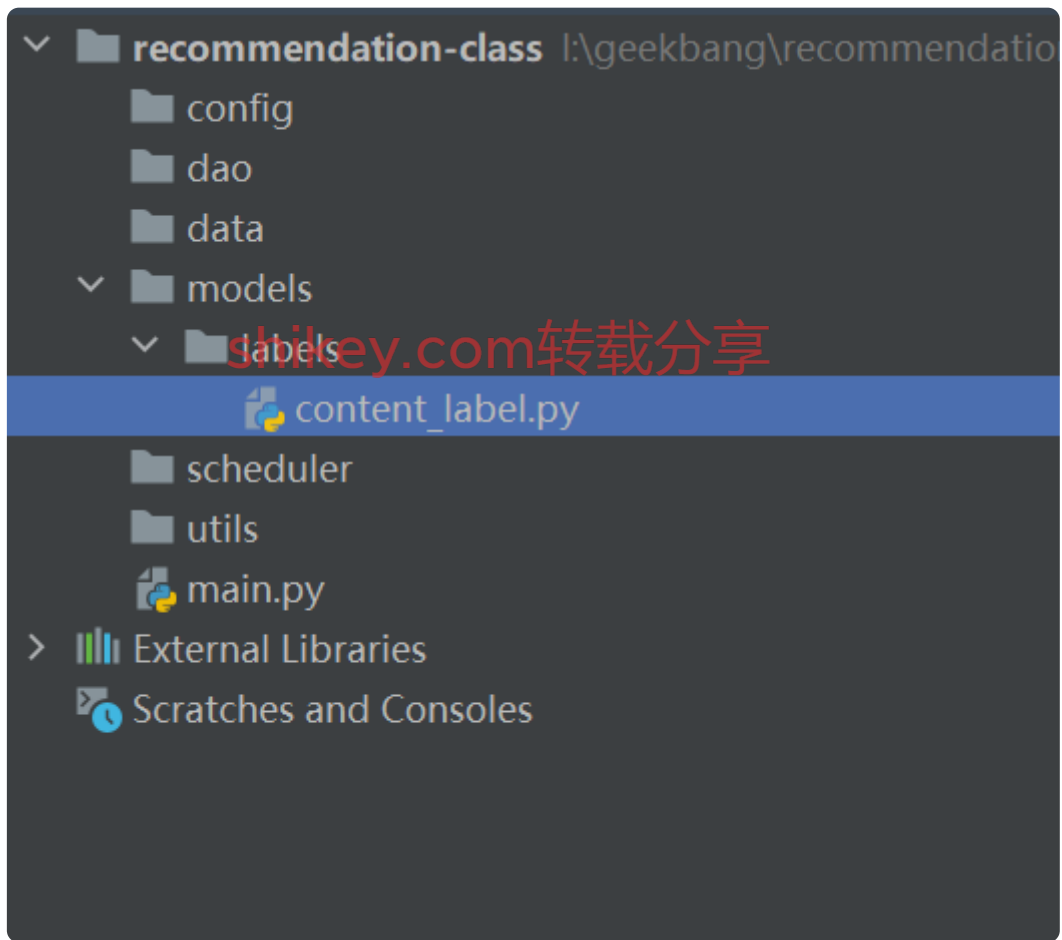
我先把最基本的一级目录列出来。



我认为，一个最基本的推荐系统目录至少要包含表格里的这六个部分。

目录	作用
config	配置文件目录，装着和推荐系统相关的所有配置文件信息
dao	数据库访问接口目录，包含了所有和数据库配置、连接相关的信息
data	数据文件目录，一般产生的训练数据或模型文件数据都放在这里
models	模型文件目录，这里的模型指的是我们的算法代码或者和主体逻辑相关的代码
scheduler	定时器文件目录，需要每隔一段时间跑一次的离线模型会放在这里
utils	工具包，这里包含了所有需要用到的一些小的工具类

接下来要做的是制作一个内容画像，我在 models 目录下建立一个 label 目录，专门来放各种画像，然后再在 lables 目录建立一个 content_label.py 文件，专门用来编写处理与内容画像相关的代码，最后目录结构如下。



接下来我们要做的就是在这里面处理内容画像，处理的简单流程如下。

1. 从 MongoDB 中获取数据。
2. 将获取到的数据进行取关键词、字数获取、其他信息获取的操作。
3. 给内容画像设置一个默认的点赞、收藏、阅读的数量。
4. 设置一个默认热度，以后可以做热度改变。
5. 将这些内容做成画像，插入到 MongoDB 数据库，数据库的 collection 为 “content_labels”。

下面我们就来一步一步实现它。

首先，从 MongoDB 中导入数据，建立数据库连接。建立数据库连接的代码实际上和 scrapy 里面连接 MongoDB 的代码是一套，所以可以直接把 scrapy 项目中 dao 目录下的 mongo_db.py 文件直接拿过来，然后根据目前的项目，稍微做一下改动，代码如下。

```

1 import pymongo
2
3 class MongoDB(object):
4     def __init__(self, db):
5         mongo_client = self._connect('localhost', 27017, '', '', db)
6         self.db_scrapy = mongo_client['scrapy_data']
7         self.db_recommendation = mongo_client['recommendation']
8
9     def _connect(self, host, port, user, pwd, db):
10        mongo_info = self._splicing(host, port, user, pwd, db)
11        mongo_client = pymongo.MongoClient(mongo_info, connectTimeoutMS=12000, co
12        return mongo_client
13
14    @staticmethod
15    def _splicing(host, port, user, pwd, db):
16        client = 'mongodb://' + host + ":" + str(port) + "/"
17        if user != '':
18            client = 'mongodb://' + user + ':' + pwd + '@' + host + ":" + str(port) + "/"
19        if db != '':
20            client += db
21        return client

```

可以看到，在这里我只是做了一个非常小的改动，就是在最上边的数据库中添加了一个“recommendation”。后面所有在推荐系统中涉及 MongoDB 数据库的操作，都会基于“recommendation”这个库来操作。但是我仍然保留了“scrapy_data”这个库，因为读取原始数据还是要基于这个库来读取。

有了这个 MongoDB 连接类，就可以将其导入到 content_label.py 文件中，然后直接在我们的 content_label.py 文件中使用。

接下来，我们来看 content_label.py 这个文件的代码。我们先来做一个简单的内容画像，其画像的字段全部来自已知数据。

```

1 from dao.mongo_db import MongoDB
2 from datetime import datetime
3
4 class ContentLabel(object):
5     def __init__(self):

```

```


6         self.mongo_scrapy = MongoDB(db='scrapy_data')
7         self.mongo_recommendation = MongoDB(db='recommendation')
8         self.scrapy_collection = self.mongo_scrapy.db_scrapy['content_ori']
9         self.content_label_collection = self.mongo_recommendation.db_recommendatio
10
11     def get_data_from_mongodb(self):
12         datas = self.scrapy_collection.find()
13         return datas
14
15     def make_content_labels(self):
16         datas = self.get_data_from_mongodb()
17         for data in datas:
18             content_collection = dict()
19             content_collection['describe'] = data['desc']
20             content_collection['type'] = data['type']
21             content_collection['title'] = data['title']
22             content_collection['news_date'] = data['times']
23             content_collection['hot_heat'] = 10000
24             content_collection['likes'] = 0
25             content_collection['read'] = 0
26             content_collection['collections'] = 0
27             content_collection['create_time'] = datetime.utcnow()
28             print(content_collection)
29
30 if __name__ == '__main__':
31     content = ContentLabel()
32     content.make_content_labels()

```

shikey.com转载分享

这是一个很简单的画像的例子，当然，目前我还没有把它们插入到 MongoDB 数据库中，仅仅是使用 print() 函数将它们打印了出来。


我们来稍微解析一下这段代码。在这段代码中首先在上面引入了 MongoDB 的连接类，又引入了一个 datetime 库，你可以利用这个库来获取到当前时间并赋值到创建时间这个字段中。

 复制代码

```
1 content_collection['create_time'] = datetime.utcnow()
```

在 __init__() 函数的定义中，我们主要是定义了与数据库连接相关的变量，主要有 MongoDB 连接 scrapy 库、MongoDB 连接 recommendation 库（注意，这个库目前数据库里还没有，需要在数据库手动创建），以及对应的两个 collection，分别是 “scrapy_data” 库的


“content_ori” 和 “recommendation” 库的 “content_label” （这个 collection 目前也没有，当运行程序后会自动创建）。

 复制代码

```
1 self.scrapy_collection = self.mongo_scrapy.db_scrapy['content_ori']
2     self.content_label_collection = self.mongo_recommendation.db_recommendatio
```

shikey.com 转载分享

紧接着，我们创建了一个名为 “get_data_from_mongodb” 的函数，这个函数主要是从 scrapy 库中获取到原始数据，然后将原始数据进行返回。

 复制代码

```
1     def get_data_from_mongodb(self):
2         datas = self.scrapy_collection.find()
3         return datas
```

你可以看到，在这个函数里目前虽然只有一行数据，但是我还是单独给作为一个函数拿了出来。这样做的好处是能够很好地进行解耦，这个函数就是用作获取原始数据，职责单一。

当然，因为目前数据量比较少，所以我是直接获取了所有的数据。但是当数据量非常大的时候（比如有几万或者几十万甚至上百万数据），必须采用分页的形式来获取数据，这样可以减少数据库的开销以及程序卡顿。

然后到了内容画像中最重要的一个函数：make_content_labels() 函数。我们会在这个函数中进行数据的组装，并把组装后的数据作为内容画像存储在 MongoDB 数据库中。

MongoDB 所使用的数据格式叫 BSON，BSON 是一种类似于 JSON 的数据类型。在 Python 中，Dict 类型和 JSON 类型可以相互转换。因此我们在这里新建了一个 Dict() 类型的变量，名为 “content_collection”，然后再将所需要的内容给填充进去。我们首先从原始数据中获取到基本的信息，例如内容、类型、标题和新闻时间，然后放入到字典中。

 复制代码

```
1     def make_content_labels(self):
2         datas = self.get_data_from_mongodb()
```

```
3     for data in datas:
4         content_collection = dict()
5         content_collection['describe'] = data['desc']
6         content_collection['type'] = data['type']
7         content_collection['title'] = data['title']
8         content_collection['news_date'] = data['times']
```

shickey.com转载分享

接着，我又新建了一些初始化的特征。例如在这里我设定了初始化的热度为 10000，这是为了以后可以通过热度值来进行排序。然后我又新建了点赞、阅读、收藏这三个值，初始值为 0。最后我又定义了一个创建时间作为这条数据的创建时间，它可以用作后面一些模型和算法的特征。

复制代码

```
1     content_collection['hot_heat'] = 10000
2     content_collection['likes'] = 0
3     content_collection['read'] = 0
4     content_collection['collections'] = 0
```


然后，我使用 print 先将数据打印出来看一下，得到如下结果。



如果出现这个结果，至少说明我们的程序目前是可以正常跑通的。接下来，我们就可以再向里面加点东西。

我们可以往里面再加入一个统计字数的功能。在 Python 中，字数可以使用正则化加上 Unicode 编码的方式来进行统计。我们知道，实际上汉字也是 Unicode 编码的一部分。在

Unicode 编码中，常用汉字的编码是从 4E00 到 9FA5，因此将这一部分的内容统计出来即可。你可以在程序中引入 re 库来进行正则表达式的统计，这段函数可以写成下面的形式。

 复制代码

```
1 def get_words_nums(self, contents):  
2     ch = re.findall('([\u4e00-\u9fa5])', contents)  
3     num = len(ch)  
4     return num
```

shike.com 转载分享


在这段函数中传入的是文本内容，也就是我们爬取到的文章正文，然后使用 re.findall() 函数来取得内容中 Unicode 编码在 4E00 到 9FA5 之间的文字。这个时候会得到一个 list 列表，列表里面只包含常用的中文文字，之后用 len() 函数对这些文字的数量进行统计，就得到了需要的中文字数。

然后，我们在 make_content_labels() 中加入它的函数调用即可。

 复制代码

```
1 content_collection['keywords'] = keywords
```

现在我们可以加上 MongoDB 插入的部分，看看能不能正常写入到数据库中。这一步其实非常简单，只需要在 make_content_labels() 函数的最后加入下面这行代码即可。

 复制代码

```
1 self.content_label_collection.insert_one(content_collection)
```

我们来运行一下这段程序，运行后刷新一下 MongoDB 数据库，就得到了如下界面。



到这里就说明最基础的画像已经能够正常写入到 MongoDB 数据库了，在接下来的课程中我们会继续完善这个内容画像，并将其作为模型和算法的特征。

实际上这里还有一个非常关键的特征没有加上，那就是关键词。也就是提取文章和标题中的关键词，然后再加入到我们的画像中作为画像的一部分。这一部分的内容我想留作一个作业让你课后去完成，我会在 GitHub 里公布我们的代码和思路。

总结

我们来总结一下这节课的内容，学完这节课，希望你能够记住以下要点。

1. 知道内容画像是什么，它在推荐系统中有着举足轻重的地位。
2. 了解非结构化文本内容画像的生成处理方式，比如文本分类、文本聚类、关键词提取等等。
3. 熟悉如何使用 Python 配合 MongoDB 来做一个简单的内容画像。

课后题

最后依旧是课后题环节，给你留了两道课后题。

1. 完成并理解课程中的代码，再想想还可以挖掘哪些特征并找一个特征加入到的内容画像中。
2. 在原有代码的基础上加入关键词的特征，关键词提取限定使用 TF-IDF 或者 TextRank，也可以两者结合使用。

© 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。

shikey.com转载分享

精选留言 (5)



Geek_ccc0fd

2023-05-08 来自广东

关于画像有个问题想请教一下老师：

我们训练样本一般是过去一段时间的数据，但是画像数据保存的最新的画像标签，这里如果直接使用样本关联画像标签的话会发生特征穿越问题，这里实际工作中是如何处理的呢？

作者回复：特征穿越问题主要是因为是在训练模型时，使用了一些在未来才能得到的标签或特征，导致模型在实际使用时表现不佳。在用户画像中，由于画像数据是最新的，而训练样本是过去一段时间的数据，因此会遇到特征穿越问题。

解决这个问题的方法主要有以下几种：

1. 对历史数据进行特征工程，将历史数据中的一些特征转化为对未来数据有预测能力的特征。这样就可以使用历史数据中的特征来预测未来的画像标签，避免了特征穿越问题。
2. 将画像标签作为新的训练样本特征，同时使用前面的历史数据特征作为输入，来训练模型。这种方法可以在模型中加入画像标签的影响，提高模型的预测准确性。同时，还可以使用 rolling window 等方法，避免将未来数据引入模型中。
3. 对数据进行时间切片，将过去一段时间的数据作为一个时间窗口，来训练一个对应的模型。然后使用该模型来预测下一个时间窗口的画像标签。这样就可以保证模型只使用过去的的数据，不受未来数据的影响。

综上，面对特征穿越问题，我们需要针对具体场景来选择合适的解决方案。需要考虑的因素包括数据量、数据质量、特征工程复杂度、计算资源等等。

共 5 条评论 >



1



GhostGuest

2023-05-10 来自上海

文稿中热度设置错了，代码写的一万，文稿写的一千

作者回复: 同学你好, 谢谢你的指正, 已修改。



Geek_ea1710

2023-05-09 来自陕西

已看

shikey.com转载分享



翡翠虎

2023-05-08 来自广西

除了关键词外, 我感觉文章类型 (文本分类)、国家地区也可以作为特征之一

作者回复: 是的, 这里只是举了个例子, 实际上可以有更多特征。



Geek_ccc0fd

2023-05-08 来自广东

统计字数那里赋的代码是不是搞错了

作者回复: 这个错在哪里呢, 我测试过了, 没有什么问题。

共 4 条评论 >

