周工作总结20190520-20190524

这周主要的工作是将simhash去重服务做了一些性能和效果测试并做到上线，同时给几个api添加了fast天眼预警，观测接口状态和异常的。

技术积累

1. 滑动窗口计算句子的窗口simhash值

   这是一个解决实际问题的改进点，记录一下，希望给以后一些启发。问题是这样的，我们用simhash做文本去重的时候，直接给文本计算一个hash值，只能找到长度相近的相似文章，对于截取了文章一部分的这种情况无法判断，因此提出了一种滑动窗口的机制，先对文本做滑动窗口，以14个词为一段，给这14个词计算一个hash值，然后存储，在计算相似度的时候，就看文本A和文本B有多少个重复的段。

   计算方法为：

```
def slide_window(self, line):
        """
        通过滑动窗口机制，求出句子的窗口段
        :param line:
        :return: sentence_seg 句子段的列表
        """
        sentence_seg = []
        sentence_list = line.split(" ")
        length = len(sentence_list)
        # print(length)
        if length <= 14:
            return [line]
        i = 0
        # 以14为窗口长度，5为步长，滑动
        while i < length and sentence_list[i]:
            if i+14 <= length - 1:
                sentence_seg.append(sentence_list[i: i+14])
            elif length - 14 >= 0:
                sentence_seg.append(sentence_list[length-14:
length])
                i += 14
            i += 5
        return sentence_seg
```

2. python**程序并发**

   在提取历史数据进行计算各种值的过程中，发现分词的速度比较慢，影响了整体的进度，因此就想用并发的方式来提速。崔鸣给推荐了**concurrent.futures模块**。下面是一段示例：

```
def prepare_batch(self, text_list):
        result_queue = []
        es_body_queue = []
        with ThreadPoolExecutor(max_workers=25) as executor:
            futures = [executor.submit(self.prepare_es, text)
for idx, text in
                       enumerate(text_list)]
            for f in as_completed(futures):
                result_queue.append(f.result())

        with ThreadPoolExecutor(max_workers=25) as executor:
            futures = [executor.submit(self.generate_es_body,
result) for idx, result in
                       enumerate(result_queue)]
            for f in as_completed(futures):
                es_body_queue.append(f.result())
        return es_body_queue
```

   其中1. as_completed是一个内置函数，它的作用是为了一次性将所有结果拿到。

   executor.submit()的参数分别是要并行执行的函数和函数执行需要的参数。

3. shell查看自己目录下各目录占用空间：

```
du -h --max-depth=1 ./
```

4. urlopen在python2和python3中的差异

   差异主要体现在urlencode、request、urlopen等函数。

   在Python2中的用法:

```
import urllib2
values = {"username":"962457839@qq.com","password":"XXXX"}
data = urllib.urlencode(values)
url = "https://passport.csdn.net/account/login?
from=http://my.csdn.net/my/mycsdn"
request = urllib2.Request(url,data)
response = urllib2.urlopen(request)
print response.read()
```

在Python3中的用法：

```
from urllib import request
from urllib import parse
from urllib.request import urlopen
values = {'username': '962457839@qq.com', 'password': 'XXXX'}
data = parse.urlencode(values).encode('utf-8')  # 提交类型不能为
str，需要为byte类型
url = 'https://passport.csdn.net/account/login?
from=http://my.csdn.net/my/mycsdn'
request = request.Request(url, data)
response = urlopen(request)
print(response.read().decode())
```

5. 自动生成和安装requirements依赖

   生成requirements.txt文件

   pip freeze > requirements.txt

   安装requirements.txt依赖

   pip install -r requirements.tx

6. Ast_literal_eval()函数

   将文件中按行存储的list字符串读成list类型

```
import ast
with open('filename.txt', 'r') as f:
    mylist = ast.literal_eval(f.read())
```

异常问题

这周遇到的异常都转化成知识点放在上面了。

个人反思

1. 这周比较积极的地方是主动比较了文彬给我的线上接口和测试接口的结果，发现了一个问题，避免了去重接口的一个风险。说明最近这一段时间的积累有点成效了。在实际工作的时候还是需要**主动去发现问题**，解决问题，才能不断优化工作内容，取得更好的结果，包括之前遇到的接口访问时间，并发执行这些都是可以改进的点。
2. 这周不好的地方是基本的工作是完成了，但是进阶的一些没有做，像是把崔鸣写的ssc服务的配置文件模块重构，给离线写es数据流增加多种增删的方式，不只是能读kafka，还应该能直接从mysql取历史数据并计算然后写入es，还有能根据某些字段删数据等等，这些工作是**对已有工作的提升**，也应该在完成任务的同时一并完成的，这些才是工作能力的体现，只是完成并不算什么。