

日志分析

概述

生成中会生成大量的系统日志、应用程序日志、安全日志等等日志，通过对日志的分析可以了解服务器的负载、健康状况，可以分析客户的分布情况、客户的行为，甚至基于这些分析可以做出预测。

一般采集流程

日志产出 -> 采集 (Logstash、Flume、Scribe) -> 存储 -> 分析 -> 存储 (数据库、NoSQL) -> 可视化

开源实时日志分析ELK平台

Logstash收集日志，并存放到ElasticSearch集群中，Kibana则从ES集群中查询数据生成图表，返回浏览器端

数据提取

半结构化数据

日志是半结构化数据，是有组织的，有格式的数据。可以分割成行和列，就可以当做表理解和处理了，当然也可以分析里面的数据。

文本分析

日志是文本文件，需要依赖文件IO、字符串操作、正则表达式等技术。

通过这些技术就能够把日志中需要的数据提取出来。

```
183.60.212.153 - - [19/Feb/2013:10:23:29 +0800] "GET /o2o/media.html?menu=3 HTTP/1.1" 200 16691  
"- " "Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)"
```

这是最常见的日志，nginx、tomcat等WEB Server都会产生这样的日志。如何提取出数据？

这里面每一段有效的数据对后期的分析都是必须的。

提取数据代码实现

```
with open('xxx.log') as f:  
    for line in f:  
        for field in line.split():  
            print(field)
```

缺点：

数据并没有按照业务分割好，比如时间就被分开了，URL相关的也被分开了，User Agent的空格最多，被分割了。所以，定义的时候不选用这种在filed中出现的字符就可以省很多事，例如使用'\x01'这个不可见的ASCII，
print('\x01')试一试

使用正则表达式

```
import re

line = '''183.60.212.153 - - [19/Feb/2013:10:23:29 +0800] \
"GET /o2o/media.html?menu=3 HTTP/1.1" 200 16691 "-" \
"Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)"""

pattern = '([\d.]{7,}) - - \[[\w/: +]\] "(\w+) (\S+) ([\w\d/.]+)" (\d+) (\d+) ".+" '
(.*?)'''
regex = re.compile(pattern)

def extract(line:str):
    m = regex.match(line)
    if m:
        print(m.groups())

extract(line)
```

使用上面的分组就可以提取到所有的分组，就是我们想要的数据。

类型转换

fields中的数据是有类型的，例如时间、状态码等。对不同的field要做不同的类型转换，甚至是自定义的转换

时间转换

19/Feb/2013:10:23:29 +0800 对应格式是 %d/%b/%Y:%H:%M:%S %z

使用的函数是datetime类的strptime方法

```
import datetime

def convert_time(timestr):
    return datetime.datetime.strptime(timestr, '%d/%b/%Y:%H:%M:%S %z')
```

可以得到

```
lambda timestr: datetime.datetime.strptime(timestr, '%d/%b/%Y:%H:%M:%S %z')
```

状态码和字节数

都是整型，使用int函数转换

映射

对每一个字段命名，然后与值和类型转换的方法对应。

最简单的方式，就是使用正则表达式分组。

```
import datetime
import re

line = '''183.60.212.153 - - [19/Feb/2013:10:23:29 +0800] \
"GET /o2o/media.html?menu=3 HTTP/1.1" 200 16691 "-" \
"Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)"""
```

```

pattern = '''(?P<remote>[\d.]{7,}) - - \[(?P<datetime>[\w/: +]+\)\] \
" (?P<method>\w+) (?P<url>\S+) (?P<protocol>[\w\d/.]+)" (?P<status>\d+) (?P<length>\d+) \
".+" "(?P<useragent>.+)"'''
regex = re.compile(pattern)

conversion = {
    'datetime': lambda timestr: datetime.datetime.strptime(timestr, '%d/%b/%Y:%H:%M:%S %z'),
    'status': int,
    'length': int
}

def extract(logline: str) -> dict:
    m = regex.match(line)
    if m:
        return {k:conversion.get(k, lambda x:x)(v) for k,v in m.groupdict().items()}

print(extract(line))

```

异常处理

日志中不免会出现一些不匹配的行，需要处理。

这里使用re.match方法，有可能匹配不上。所以要增加一个判断

采用抛出异常的方式，让调用者获得异常并自行处理。

```

def extract(logline: str) -> dict:
    """返回字段的字典，抛出异常说明匹配失败"""
    m = regex.match(line)
    if m:
        return {k:conversion.get(k, lambda x:x)(v) for k,v in m.groupdict().items()}
    else:
        raise Exception('No match. {}'.format(line)) # 或输出日志记录

```

但是，也可以采用返回一个特殊值的方式，告知调用者没有匹配。

```

def extract(logline: str) -> dict:
    """返回字段的字典，如果返回None说明匹配失败"""
    m = regex.match(line)
    if m:
        return {k:conversion.get(k, lambda x:x)(v) for k,v in m.groupdict().items()}
    else:
        return None # 或输出日志记录

```

通过返回值，在函数外部获取了None，同样也可以采取一些措施。本次采用返回None的实现。

数据载入

对于本项目来说，数据就是日志的一行行记录，载入数据就是文件IO的读取。将获取数据的方法封装成函数。

```
def load(path):  
    """装载日志文件"""  
    with open(path) as f:  
        for line in f:  
            fields = extract(line)  
            if fields:  
                yield fields  
            else:  
                continue # TODO 解析失败就抛弃，或者打印日志
```