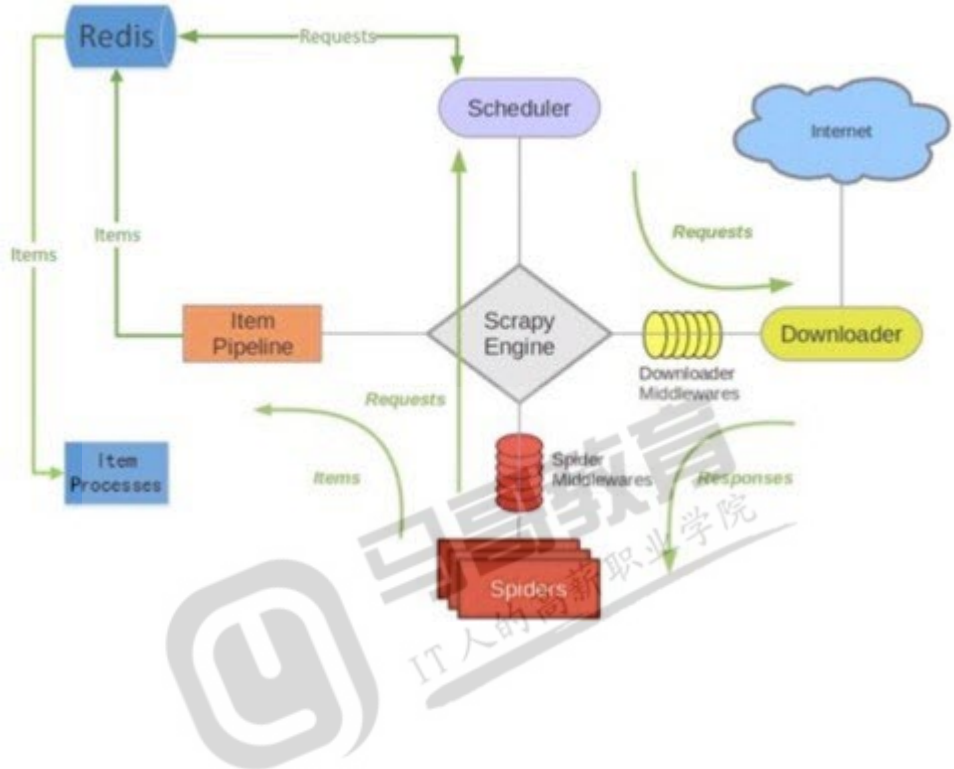
Scrapy-redis组件

概述

这是一个能给Scrapy框架引入分布式的组件。
分布式由Redis提供，可以在不同节点上运行爬虫，共用同一个Redis实例。
在Redis中存储待爬取的URLs、Items。



安装

```
$ pip install scrapy-redis
```

安装要求
Python 2.7, 3.4 or 3.5
Redis >= 2.8
Scrapy >= 1.0
redis-py >= 2.10

配置

```
# Enables scheduling storing requests queue in redis.
SCHEDULER = "scrapy_redis.scheduler.Scheduler"

# Ensure all spiders share same duplicates filter through redis.
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"

# Store scraped item in redis for post-processing.
ITEM_PIPELINES = {
    'scrapy_redis.pipelines.RedisPipeline': 300
}
```

```
# The item pipeline serializes and stores the items in this redis key.
# 这个key很重要
#REDIS_ITEMS_KEY = '%(spider)s:items'

# Specify the host and port to use when connecting to Redis (optional).
#REDIS_HOST = 'localhost'
#REDIS_PORT = 6379

# Default start urls key for RedisSpider and RedisCrawlSpider.
#REDIS_START_URLS_KEY = '%(name)s:start_urls'
```

在以下方面做了增强

Scheduler + Duplication Filter, Item Pipeline, Base Spiders

- Scheduler
  本质上将原来的普通队列，变成了redis以提供多爬虫多进程共享，并行能力增强。
- Duplication Filter
  scrapy使用set来去重，scrapy-redis使用redis的set类型去重
- Item Pipeline
  在Item Pipeline增加一个处理，即将数据items存入redis的items queue中
- Base Spiders
  提供了使用了RedisMixin的RedisSpider和RedisCrawlSpider，从Redis中读取Url。

Redis是服务，爬虫就是它的客户端，客户端就可以扩展出并行的很多爬虫一起爬取。

redis安装

这里不再赘述。

豆瓣影评分析项目

抓取内容分析

抓取最新top 1电影，分析其影评

点击“全部正在热映”，跳转至`https://movie.douban.com/cinema/nowplaying/beijing/`。

这部分内容是直接通过网页HTML返回的，提取影片id的xpath为`//div[@id="nowplaying"]//li[1]/@id`

影片页

点击电影，出现影片主题页面`https://movie.douban.com/subject/26636712/`，在页面下面的短评处点击"全部(n)条"进入影片评论页面，从而得到影评的链接`https://movie.douban.com/subject/26636712/comments?start=20&limit=20`

测试发现，只有start有用，limit不能控制返回的条目数

start测试到220时，发现不能返回数据了。其实很多网站都有这种策略，显示的数据可能有很多页，但是人一般不可能看那么多页，能看查回来的10页结果就不错了。

提取影评的xpath

//div[@class="comment-item"]//span[@class="short"]

创建Scrapy项目

$ scrapy startproject review moviereview

配置

使用scrapy-redis，配置如下

```
BOT_NAME = 'review'

SPIDER_MODULES = ['review.spiders']
NEWSPIDER_MODULE = 'review.spiders'

USER_AGENT = "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/55.0.2883.75 Safari/537.36"

ROBOTSTXT_OBEY = False
DOWNLOAD_DELAY = 3
COOKIES_ENABLED = False

# Enables scheduling storing requests queue in redis.
SCHEDULER = "scrapy_redis.scheduler.Scheduler"

# Ensure all spiders share same duplicates filter through redis.
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"

# Don't cleanup redis queues, allows to pause/resume crawls.
#SCHEDULER_PERSIST = True

# The item pipeline serializes and stores the items in this redis key.
#REDIS_ITEMS_KEY = '%(spider)s:items'

#ITEM_PIPELINES = {
#    'review.pipelines.ReviewPipeline': 300,
#    'scrapy_redis.pipelines.RedisPipeline': 300
#}

# Specify the host and port to use when connecting to Redis (optional).
REDIS_HOST = 'localhost'
REDIS_PORT = 6379

#LOG_LEVEL = 'DEBUG'
```

构建Item

```
import scrapy

class ReviewItem(scrapy.Item):
    review = scrapy.Field()
```

构建爬虫

$ scrapy genspider -t crawl dbreview douban.com

写完先测试，然后将类型改为RedisCrawlSpider

```
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule
from ..items import ReviewItem
from scrapy_redis.spiders import RedisCrawlSpider

class DbreviewSpider(RedisCrawlSpider): # scrapy-redis的类
    name = 'dbreview'
    allowed_domains = ['douban.com']

    #start_urls = ['https://movie.douban.com/subject/26636712/comments?start=0&limit=20']
    """Spider that reads urls from redis queue (myspider:start_urls)."""
    redis_key = 'dbreview:start_urls'

    rules = (
        Rule(LinkExtractor(allow=r'start=\d+'), callback='parse_item', follow=False),
    )


    def parse_item(self, response):
        print('-'*30)

        comment = '//div[@class="comment-item"]//span[@class="short"]/text()'
        reviews = response.xpath(comment).extract()
        for review in reviews:
            item = ReviewItem()
            item['review'] = review.strip()
            yield item
```

爬取

`$ scrapy crawl dbreview`

会发现程序会卡住，这是因为在等待起始URL

手动添加开始url

redis中

```
lpush dbreview:start_urls https://movie.douban.com/subject/26636712/comments?start=0&limit=20
```

分析

使用爬虫，爬取所有数据，然后使用redis中的数据开始分析

jieba分词

安装 `pip install jieba`

官网 https://github.com/fxsjy/jieba

测试代码

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 精确模式

seg_list = jieba.cut("他来到了网易杭研大厦")  # 默认是精确模式
print(", ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print(", ".join(seg_list))

s = jieba.lcut("他来到了网易杭研大厦") # 直接返回列表
print(s)

s = jieba.cut("他来到了网易杭研大厦") # 返回生成器
print(s)
```

stopword 停用词

数据清洗：把脏数据洗掉。检测出并去除掉数据中无效或无关的数据。例如，空值、非法值的检测，重复数据检测等。

对于一条条影评来说，我们分析的数据中包含了很多无效的数据，比如标点符号、英文的冠词、中文"的"等等，需要把它们清除掉。
使用停用词来去除这些无效的数据。

wordcloud词云

https://amueller.github.io/word_cloud/index.html

依赖numpy、matplotlib

```
pip install wordcloud
```

常用方法

方法	说明
fit_words(frequencies)	Create a word_cloud from words and frequencies.
generate(text)	Generate wordcloud from text.
generate_from_frequencies(frequencies[, ...])	Create a word_cloud from words and frequencies
generate_from_text(text)	Generate wordcloud from text
process_text(text)	Splits a long text into words, eliminates the stopwords
recolor([random_state, color_func, colormap])	Recolor existing layout
to_array()	Convert to numpy array
to_file(filename)	Export to image file

```
from redis import Redis
import json
import jieba


redis = Redis()
stopwords = set()
with open('chineseStopWords.txt', encoding='gbk') as f:
    for line in f:
        print(line.rstrip('\r\n').encode())
        stopwords.add(line.rstrip('\r\n'))
print(len(stopwords))
print(stopwords)
items = redis.lrange('dbreview:items', 0, -1)
print(type(items))



words = {}
for item in items:
    val = json.loads(item)['review']
    for word in jieba.cut(val):
        words[word] = words.get(word, 0) + 1

print(len(words)) # 829
print(sorted(words.items(), key=lambda x:x[1], reverse=True))
#[(' , ', 119), ('的', 73), ('。', 55), ('了', 42), ('是', 23), (' ', 22), ('人', 19),
# ('也', 19), ('和', 16), ('彩蛋', 16), ('！', 15), ('反派', 13),
# ('蚁', 13), ('在', 12), ('我', 12), ('都', 12), ('被', 11), ('很', 11), ('好', 10)

words = {}
for item in items:
```

```
    val = json.loads(item)['review']
    for word in jieba.cut(val):
        if word not in stopwords:
            words[word] = words.get(word, 0) + 1


total = len(words)
print(total)
frenq = {k:v/total for k,v in words.items()}
print(sorted(frenq.items(), key=lambda x:x[1], reverse=True))

from wordcloud import WordCloud
import matplotlib.pyplot as plt

wordcloud = WordCloud(font_path='simhei.ttf', background_color='white',
                      max_font_size=80)



plt.figure(2)
wordcloud.fit_words(frenq)
plt.imshow(wordcloud)
plt.axis('off') # 去掉坐标系
plt.show()
```