

数据仓库课程报告

ETL练习——获取电影数量

指导教师: 朱宏明

谭梓煊 (1853434)

刘文朔 (1851008)

Data Warehouse

Tongji University

School of Software Engineering

目录

- 一、项目概述
- 二、数据获取
 - A. 亚马逊电影评论数据集(开源)
 - B. 亚马逊25万商品数据(爬虫)
 - 爬虫框架
 - 反反爬虫
 - 页面解析
 - 最终结果
- 三、数据处理
 - 1. 开源评论数据集 -> 商品asin码列表
 - 方案
 - 注意事项
 - 代码实现
 - 输出
 - 2. 原始商品数据 -> 电影数据
 - 判断标准
 - 进阶方法
 - 3. 电影数据 -> 电影节点, 关系列表
 - 实现思路
 - 运行结果
 - 4. 节点, 关系列表 -> 图数据库
- 四、总结

一、项目概述

- 近年来, 电影、DVD、歌剧等影像作品成为了人们生活中不可或缺的休闲消遣之物, 自电影于20世纪初成规模时起, 在荧幕上出现过的影视作品数不胜数, 让想要从浩如烟海的影视作品中筛选出自己需要的内容人们也变得举步维艰, 因此我们设想设计一个能够支持复杂条件查询, 大数据的电影信息系统
- 在本次练习中, 我们使用Scrapy框架爬取了25万亚马逊商品页面, 使用Python脚本进行数据清洗和处理, 筛选出约20万电影数据, 提取出电影信息和电影之间的关联, 使用neo4j图数据库存储电影节点和电影之间的同类关系, 调用neo4j内置的图算法求解弱连通分量个数, 最终得到不同电影的总数约为12万
- 在爬虫设计中, 我们使用 [ProxyBroker](#) 工具作为ip代理池, 使用 [fake-useragent](#) 随机切换浏览器UA, 并在此基础上通过设置请求速度等方式来突破Amazon的反爬虫机制。使用xpath和正则表达式来解析爬取到的html、提取商品信息, 使用 [jsonlines](#) 格式存储网页解析后的数据
- 在数据清洗流程中, 我们通过解析商品信息的几个特定属性进行判断, 尽可能地将非电影的商品信息去除而将电影的商品信息保留
- 最终, 我们将剩余信息按照节点和关系分别导出为CSV文件, 使用neo4j图数据库来处理信息

二、数据获取

A. 亚马逊电影评论数据集(开源)

- 亚马逊电影评论数据来自SNAP发布的 [Web data: Amazon movie reviews](#) 数据集. 该数据集包含了跨度超过10年, 总数超过8百万条的亚马逊电影评论. 每条评论包括商品信息、用户信息、评分和纯文本格式的评论内容.
- 该数据集采用纯文本格式保存, 数据格式如下:

```
1 | product/productId: B00006HAXW
2 | review/userId: A1RSDE90N6RSZF
3 | review/profileName: Joseph M. Kotow
4 | review/helpfulness: 9/9
5 | review/score: 5.0
6 | review/time: 1042502400
7 | review/summary: Pittsburgh - Home of the OLDIES
8 | review/text: I have all of the doo wop DVD's and this one is as good or
9 | better than the
10 | 1st ones. Remember once these performers are gone, we'll never get to see
11 | them again.
12 | Rhino did an excellent job and if you like or love doo wop and Rock n Roll
13 | you'll LOVE
14 | this DVD !!
```

- 数据集统计信息

Catagory	value
Number of reviews	7,911,684
Number of users	889,176
Number of products	253,059
Users with > 50 reviews	16,341
Median no. of words per review	101
Timespan	Aug 1997 - Oct 2012

B. 亚马逊25万商品数据(爬虫)

- 在亚马逊电影评论数据集中, 提取出所有商品asin码(见下文"数据处理1")之后, 即可开始构造爬虫爬取商品数据.

爬虫框架

- 我们使用了Scrapy框架进行数据爬取工作
- Scrapy框架包含了从请求发送到数据解析, 存储的整套流程, 具有强大的可定制性和可拓展性, 让开发者可以从复杂的错误处理, 频率控制等操作中脱离出来, 更加关注于整体流程.

反反爬虫

- 本次项目使用开源的 [ProxyBroker](#) 作为ip代理池
 - ProxyBroker可以自动收集网络上可用的ip代理, 并启动一个本地的代理服务器, 将请求转发至这些ip代理处

```
~ proxybroker serve --host 127.0.0.1 --port 8888 --types HTTP HTTPS --lvl High

~ export HTTP_PROXY=http://127.0.0.1:8888; export HTTPS_PROXY=http://127.0.0.1:8888
~ http http://httpbin.org/get?show_env
```

- 使用 [fake-useragent](#) 随机生成浏览器UserAgent
 - 将fake-useragent以中间件的形式集成到Scrapy框架中

```
1 # @spider/amaspd/middlewares/UserAgentMiddleware.py
2
3 from fake_useragent import UserAgent
4
5 class UserAgentMiddleware:
6
7     @classmethod
8     def from_crawler(cls, crawler):
9         # This method is used by Scrapy to create your spiders.
10        return cls(crawler.settings)
11
12    def __init__(self, settings):
13        self.ua = UserAgent()
14
15    def process_request(self, request, spider):
16        request.headers['User-Agent'] = self.ua.random
```

- 然后在Scrapy的settings.py中添加中间件

```
1 DOWNLOADER_MIDDLEWARES = {
2     'amaspd.middlewares.UserAgentMiddleware.UserAgentMiddleware': 502,
3 }
```

- 在请求头中添加一些header, 使得请求更像是浏览器发出的

```

1 headers = {
2     'Accept':
3     'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
4     'Accept-Encoding': 'gzip, deflate, br',
5     'Accept-Language': 'en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7,zh-
6     TW;q=0.6',
7 }
8
9 # @settings.py
10 DEFAULT_REQUEST_HEADERS = {
11     'Upgrade-Insecure-Requests': 1,
12     'Pragma': 'no-cache',
13     'Cache-Control': 'no-cache',
14 }

```

- 禁用Cookie

```
1 COOKIES_ENABLED = False
```

- 忽略Robot.txt

```
1 ROBOTSTXT_OBEY = False
```

- 控制发送请求的速度和频率, 设置随机等待时间

```

1 CONCURRENT_REQUESTS = 16
2 DOWNLOAD_DELAY = 0.1
3 CONCURRENT_REQUESTS_PER_DOMAIN = 16
4 CONCURRENT_REQUESTS_PER_IP = 16

```

页面解析

- 在爬取过程中, 我们发现Amazon电影页面整体分为两种样式, 一种为黑色背景的Prime Video页面, 另一种为白色背景的一般商品页. 对于这两种页面我们采用了不同的解析方法.

1. 对于Prime Video页面, 我们提取了如下三处信息

1. html文档head标签中, 具有name="title"属性的meta标签信息

```
<meta name="title" content="Watch Colosseum - Rome's Arena of Death | Prime Video">
```

- 对应解析代码

```
1 title = response.xpath('//meta[@name="title"]/@content').extract()[0]
```

2. 页面左侧的商品基本信息

Starring	Jamel Aroui, Lotfi Dziri, Derek Lea
Genres	Documentary
Subtitles	English [CC]
Audio languages	English

- 对应解析代码

```

1 primeMeta = {}
2 dts = response.xpath('//div[@id="meta-info"]//dl/dt')
3 for dt in dts:
4     key = ''.join(dt.xpath('..//text()').extract())
5     value = ''.join(dt.xpath('..//dd//text()').extract())
6     primeMeta[key] = value

```

3. 页面底部的Other formats列表

Other formats

DVD
from \$7.00

- 对应解析代码

```

1 otherFormat = []
2 otherFormatHrefs = response.xpath('//div[@data-automation-id="other-
  formats"]//a/@href').extract()
3 for otherFormatHref in otherFormatHrefs:
4     asin = re.search('/dp/(\w+)/', otherFormatHref).group(1)
5     otherFormat.append(asin)

```

- 示例页面解析后的结果如下

```

1 {
2     "pid": "B003VHELLI",
3     "otherFormat": [
4         "B00060BPZY"
5     ],
6     "title": "Watch Colosseum - Rome's Arena of Death | Prime Video",
7     "primeMeta": {
8         "Starring": "Jamel Aroui, Lotfi Dzirri, Derek Lea",
9         "Genres": "Documentary",
10        "Subtitles": "English [CC]",
11        "Audio languages": "English"
12    }
13 }

```

2. 对于普通页面, 我们提取了四处信息

1. html文档head标签中, 具有name="title"属性的meta标签信息

```
<meta name="title" content="Watch Colosseum - Rome's Arena of Death | Prime Video">
```

- 对应解析代码

```
1 title = response.xpath('//meta[@name="title"]/@content').extract()[0]
```

2. 页面顶部的Other formats列表
3. 页面顶部的Additional options列表

Legend of Suram Fortress [VHS]

Format: VHS Tape

DVD \$75.75	VHS Tape \$7.97				
Additional VHS Tape options		Edition	Discs	Price	New from Used from
VHS Tape		—	—	\$7.97	\$7.97 \$14.99
VHS Tape		—	1	—	— \$119.00

对应解析代码

```
1 otherFormat = []
2 otherFormatHrefs = response.xpath("//li[contains(@class,
  'swatchElement')]/a[@href!='javascript:void(0)']/@href").extract()
3 for otherFormatHref in otherFormatHrefs:
4     asin = re.search('/dp/(\w+)/', otherFormatHref).group(1)
5     otherFormat.append(asin)
6
7 additionalOptions = []
8 additionalOptionHrefs = response.xpath("//div[contains(@class, 'top-
  level')]/span/@data-tmm-see-more-editions-click").extract()
9 additionalOptionHrefs = list(filter(lambda x: '"metabindingUrl": "#" not
  in x', additionalOptionHrefs))
10 for additionalOptionHref in additionalOptionHrefs:
11     asin = re.search('/dp/(\w+)/', additionalOptionHref).group(1)
12     additionalOptions.append(asin)
```

4. 页面底部的Product details

Product details

Is Discontinued By Manufacturer : No

Package Dimensions : 7.32 x 4.19 x 1.12 inches; 6.13 Ounces

Media Format : Subtitled

ASIN : 5555223870

对应解析代码

```
1 productDetail = {}
2 detailNames =
  response.xpath('//div[@id="detailBullets_feature_div"]/ul[contains(@class
    , "detail-bullet-list")]/span[@class="a-text-bold"]')
3 for detailName in detailNames:
4     key = detailName.xpath('./text()').extract()[0][:3]
5     value = detailName.xpath('./span[last()]/text()').extract()[0]
6     productDetail[key] = value
```

示例页面解析后的结果如下

```
1 {
2     "pid": "5555223870",
3     "otherFormat": [
4         "B01M6BGD6N",
5     ],
6     "title": "Amazon.com: Legend of Suram Fortress [VHS]: Movies & TV",
7     "productDetail": {
```


最终结果

- ```

201629 [{"pid": "B000WDXGKI", "otherFormat": "[{"pid": "B0003J1SKO", "title": "Watch Hannie Caulder | Prime Video", "primeMeta": ["Directors": "Burt Kennedy", "St"}, {"pid": "B000RQFVFI", "otherFormat": [{"pid": "B0008CFFSI", "title": "Amazon.com: Mando Live Syntek, Aleks Movies & TV", "productDetail": "{ \"Is Discontinued By Manufacturer\": \"No\", \"MPAA rating"}], title": "Amazon.com: B0074TQTGR2", "otherFormat": [{"pid": "B000LXKZU2", "title": "Amazon.com: The Big Trail [VHS]: John Wayne, Marguerite Churchill, El}, {"pid": "B00029HUGA", "title": "Amazon.com: The Many Fists of Bruce Lee: Movies & TV", "productDetail": "{ \"Aspect Ratio\": \"1.33:1, 1.85:1\", \"MPAA rating\"}], {"pid": "B0077227Q8", "title": "Amazon.com: Alice Cooper: Live at Montreux, 2005 [Blu-ray]: Alice Cooper: Movies & TV", "productDetail": [{"pid": "B000LSXBQQ", "title": "Amazon.com: Pilates Core Challenge: Ana Cabn: Movies & TV", "productDetail": "{ \"Aspect Ratio\": \"1.33:1\", \"Is Discontinued By"}], {"pid": "B000SJNYJC", "title": "Amazon.com: Boy Meets World: Season 3: Ben Savage, Rider Strong, Danielle Fishel: Movies & TV", "productDetail": [{"pid": "B00APDZOZY", "title": "Watch Cornhole: The Movie | Prime Video", "primeMeta": { "...", "Genres": "Comedy", "Subtitles": "None available"}, {"pid": "I1S78730198", "title": "Amazon.com: Cracking the Code to the Extraordinary: Ramtha, Ramtha: Movies & TV", "productDetail": "{ \"MPAA rating\": \"s_medio }"], {"pid": "B001NNOVT2", "title": "Amazon.com: Tell No One [Blu-ray]: Fran\u00fcois Cluzet, Marie-Jos\u00e9 Croze, Andr\u00e9 Dussan}, {"pid": "B0001HMSPG", "title": "Amazon.com: Greyfriars Bobby: Donald Crisp, Laurence Naismith, Alice Mackenzie, Kay Walsh, Duncan}, {"pid": "B0008191VU", "title": "Amazon.com: Hawks - Ist Conflict (Vol. 1) : Tallies Jaffe, Karen Strassman: Movies & TV", "productDetail": "{ \"Is Discontinued By\"}, {"pid": "B0008AC1YV", "title": "Amazon.com: Deads of Death [VHS] Peter Cushing, John Mills, Alan Baxter, Ray Milland, Anton Diffring, Gordon Jackson, Marc}, {"pid": "B0003VSGWA", "title": "Amazon.com: Napoleon and Samantha: Michael Douglas, Bill Gear, Johnny Whitaker, James Foster, Ar}, {"pid": "I15905260AK", "title": "Amazon.com: Breakaway DVD: Stanley, Andy: Movies & TV", "productDetail": "{ \"Product Dimensions\": \"5.35 x 0.58 x 7.53 inches\"}, {"pid": "I16300156370", "title": "Amazon.com: Get Ready for Math:Wonder of Numbers [VHS] Golden Books Step Ahead videos: Movies & TV", "productDetail": "{ \"Pa\"}, {"pid": "I16302484928", "title": "Amazon.com: The Legend of Babe Ruth [VHS]: William Bendix, Claire Trevor, Charles Bickford, Sam Levene, William Franklin G}, {"pid": "I1630208915", "title": "Amazon.com: Paranoia (aka Orgasmo) (1969) [VHS]: Carroll Baker, Lou Castel, Colette Descombes, Tino Carraro, Je}, {"pid": "B0007LP516", "title": "Amazon.com: Sledge Hammer! - Season Two: David Rasche, Anne-Marie Martin, Harrison Page, Bernie Kopell, Deborah Nakeham, L}, {"pid": "B001VLBDDE", "title": "Amazon.com: The Kiss: Jack Hall, Lourdes Colon, London Lemelle, Robyn Jensen, Valerie Feuer, Annie Rachele-Haux, Michael G}, {"pid": "I1630237776", "title": "Amazon.com: Shout [VHS]: Waltraut, Travalta: Movies & TV", "productDetail": "{ ..., \"format\": \"VHS Tap\" }, {"pid": "I16300818243", "title": "Amazon.com: Animal Crackers [VHS]: Gualtero Marx, Harpo Marx, Chico Marx, The Marx Brothers, Zeppo Marx, Lillian}, {"pid": "I16301966279", "title": "Amazon.com: Bataan [VHS]: Robert Taylor, George Murphy, Lloyd Nolan, Sandra Mitchell, Les Bowman,}, {"pid": "B001VHLBEE", "title": "Amazon.com: Catch Your Mind: Megan Beale, Patrick Welsh, Margaret deAngelis, Suamy Kandan: Movies & TV", "produ}, {"pid": "B0008191VU", "title": "Amazon.com: Kol Kai, Vol. 3: The Decision: Artist Not Provided: Movies & TV", "productDetail": "{ \"Aspect Ratio\": \"1.78:1\",}, {"pid": "B005CMGN2", "title": "Amazon.com: Sister Vision Season 2 - Volume 1: Kodj Brown, Ray Schneider, Christine Brown, Jonny Brown, Merl B}, {"pid": "B000007808", "title": "Amazon.com: Roy Rogers - Grand Canyon Trail: Grey, Zane: Movies & TV", "productDetail": "{ \"MPAA rating\": \"s_mediotated NR (\"}, {"pid": "B000W950YS", "title": "Amazon.com: Life With Father [DVD]: Michael Curtiz, Clarence Day, Howard Lindsay, Russel Vrouse}, {"pid": "I16304026668", "title": "Amazon.com: Half a Loaf of Kung Fu [VHS]: Jackie Chan, Chung-Erh Lung, Jeong-Nam Kim, Chih-Pin C}, {"pid": "I159668464X", "title": "Amazon.com: Handspinning Rare Wools: Robbins, Deborah: Movies & TV", "productDetail": \"{ \"Media Format\": \"NTSC\", \"Release date\"}, {"pid": "B00011SSP6", "title": "Amazon.com: Charlie, the Lonesome Cougar: Rex Allen, Ron Brown, Edward C. Moller, Clifford Peters}, {"pid": "B0004COZGY", "title": "Amazon.com: The Complete Inspector Lewis Collection (Pilot & Full Seasons 1, 2 & 3 on 11 DVDs): Kevin Whately & Laurence Fo}, {"pid": "B000954102", "title": "Amazon.com: Real Wheels: Rockin' Real Wheels: Real Wheels: Movies & TV\", "productDetail": "{ \"Aspect Ratio\": \"1.33:1\", \"MPAA\"}, {"pid": "B002KYIAL0", "title": "Amazon.com: Ichi // The Movie [Blu-ray]: Takao Asaya, Shido Nakamura, Haruka Ayase}, {"pid": "B003E74KKK", "title": "Amazon.com: Word Is Out: Stories of Some Of Our Lives: David Gillon, Sally Gearhart, Fred Gray, Dennis Chic}, {"pid": "B003SLUNZA", "title": "Amazon.com: Baseball: The Tenth Inning:, Directed by Ken Burns and Lynn Novak: Movies & TV", "productDetail": \"{ \"Baseball: The Tenth Inning\"}, {"pid": "I1630589670", "title": "Amazon.com: The Brigitte Bardot Collection (Come Dance With Me / Please Don't Now / Les Femmes / Naughty Girl / B}, {"pid": "B003FZWTVZ", "title": "Amazon.com: Galloping Minds Preschooler Learns Numbers and Counting with Animals: Galloping Minds!, Galloping Minds!: Movie}, {"pid": "B003M1UYVS", "title": "Amazon.com: Fighting Red / Moving Violation [Double Feature]: Stephen Norriss, Kay Layton, Eddie Albert, Johnny Chapman, Will}, {"pid": "B003RT18275", "title": "Amazon.com: Baseball: A Film by Ken Burns (Includes: The Tenth Innng): Ken Burns, Ken Burns, Lynn Novack: Movie}, {"pid": "B000GRUMMA", "title": "Amazon.com: Savage Sinema From Under down-3 dvd set: Kevin Hopkins, Mark Savage: Movies & TV\", "productDetail": \"{ \"Aspect Ra\"}, {"pid": "B003F32VCX", "title": "Amazon.com: Equality U: Various, Dave O'Brien: Movies & TV\", "productDetail": "{ ..., \"format\": \"DVD\", \"additional\"}, {"pid": "B003A3PGFH", "title": "Amazon.com: Lisa Lampanelli: Long Live The Queen: Dave Higby\", \"productDetail\": {}}, {"pid": "I16304952198", "title": "Amazon.com: Journey to the Hollow Earth [VHS]: Journey to the Hollow Earth: Movies & TV\", \"productDetail\": \"{ \"La}"]}]
```

### 三、数据处理

## 1. 开源评论数据集 -> 商品asin码列表

## 方案

- 逐行遍历movies.txt文件, 寻找开头为"product/productId:"的行, 提取后面的asin逐行写入到另一个文件中

## 注意事项

- movies.txt文件编码为"iso-8859-1", 如果编码选择不正确会导致数据错乱
- 使用集合数据结构存储已写入的asin防止重复

## 代码实现

```

1 asinSet = set()
2 with open("movies.txt", encoding="iso-8859-1") as movies, open('asin.txt',
 'w', encoding="utf-8") as f:
3 for line in movies:
4 try:
5 asin = line.split("product/productId:")[1].strip()
6 if asin not in asinSet:
7 asinSet.add(asin)
8 f.write(asin+'\n')
9 except:
10 pass
11
12 print(f"Done. {len(asinSet)} asin in total")

```

## 输出

```

1 B003AI2VGA
2 B00006HAXW
3 B00004CQT3
4 B00004CQT4
5 B006JIUN2W
6 B0078V2LCY
7 (...总计253059行, 此结果与数据集说明相符)

```

## 2. 原始商品数据 -> 电影数据

- 得到原始商品数据后, 我们首先要进行数据清洗, 将非电影的商品信息去除

### 判断标准

- 我们仔细分析了亚马逊平台上的电影商品信息, 找到了一些它们不同于非电影商品的共同点, 以此为判断依据来分析一个商品是否是电影
- 1. 电影商品以及类电影商品(如纪录片)通常含有Director属性, 而多集电视剧一类的商品通常采用多导演拍摄, 因此信息中通常没有Director属性. 因此可以观察商品信息是否包含有Director属性, 如果出现则认为是电影
- 2. 商品信息中是否出现视频长度, 如果出现而且时长位于一定范围内则认为是电影
- 3. 商品信息中是否出现MPAA分级, 若有此属性且属性满足条件(不为NotRated一类的无意义分级), 则认为该商品是电影
- 4. 部分非电影商品的标题具有明显特征, 如[VHS]为数字电视广播节目的录像, Analysis of ... technique为技术教程. 如标题为此类格式可以判断为非电影
- 5. 商品的Type/Genres属性表明了该出版物的体裁类型, 而部分属性很明显不是电影所有的, 比如VHS、PBS(公共电视台节目)、fitness、TV Talk Shows、News一类. 若出现这些类型则可以直接判断为非电影
- 6. 否则认为商品不是电影
- 实际上使用简单的if-else判断法是称不上准确的, 以上的任何一条判断法则都存在着例外, 符合的不一定是电影, 不符合的不一定不是电影. 比如有些商品是时长超长的电影合集、部分带有TV类型的商品其实也是电影. **以上判断法则只能说是一种倾向, 即符合标准的我们相信它大概率确实是一部电影**

## 进阶方法

- 可以使用以下方法来改进判断标准
  1. 可以引入权重机制代替简单的if-else法. 比如若一个商品的Type为Reality TV, 则其的分数减少, 即减少其是电影的概率, 但不直接判断其不是电影. 最终根据商品的评分进行最终判断, 位于分界线附近的可以人工判断
  2. 在检测的过程中, 若发现某些无关属性(如出版商、导演等)与商品的类型呈现强相关性, 则将其也加入权重判断中去. 比如若发现某公司出版商品全部都是电影, 则可以认为该公司比较偏爱于出版电影类商品, 若再发现该公司的商品则可以提升商品的电影权重
  3. 相关属性也是同理, 若发现有商品的分类与其基于属性的判断冲突, 则认为该属性与商品的类型的相关性并没有那么强, 可以削弱其权重
  4. 判断完成后随机抽取几部人工判断, 根据2、3的法则进行权重的加减
  5. 不断重复2~4的流程, 倾向特别大的可以直接判断, 根据判断结果改变权重, 依次循环, 直至剩余商品少到可以人工判断为止
  6. 如有条件可以构建神经网络模型, 使用Classifier分类器来进行判断

经过筛选, 最终得到了 201,368 条判断为电影的数据

## 3. 电影数据 -> 电影节点, 关系列表

- 在数据清洗之后, 我们得到了判断为电影的商品数据, 接下来要从这些数据中提取出电影节点和电影之间的关系.

### 实现思路

1. 提取节点
  - 遍历所有电影的 otherFormat 和 additionalOptions 属性, 将未出现过的电影的asin提取出来, 和电影的名字一起逐行存储到一个csv文件中
  - 如果遇到不在最初25万商品数据集中的节点, 则将标题设为"others"
2. 提取关系
  - 遍历所有电影的 otherFormat 和 additionalOptions 属性, 将每一条关系逐行存储到一个csv文件中

### 运行结果

- 节点列表

- otherFormat关系列表

- additionalOptions关系列表

|        |                       |
|--------|-----------------------|
|        | additionalOptions.csv |
| 552724 | 6304026668,630402617X |
| 552725 | 6304026668,6304725019 |
| 552726 | 6304026668,B00004REK7 |
| 552727 | 6304026668,B00004CJ6H |
| 552728 | B0001I55P6,B01I067V0M |
| 552729 | B0001I55P6,B00003L9BZ |
| 552730 | B0009S4IO2,B01M68CQZ9 |
| 552731 | B0009S4IO2,B00UGQ7Q82 |
| 552732 | B0009S4IO2,B01GWHF7K  |
| 552733 | B0009S4IO2,B01M5ISP7P |
| 552734 | B003E74KRK,B00UGPUGPI |
| 552735 | B003E74KRK,B01I06GFB8 |
| 552736 | B003E74KRK,B01M4QNMZD |
| 552737 | B003S1UNZA,B0045VU87A |
| 552738 | B003S1UNZA,B01M7X2J80 |
| 552739 | B003S1UNZA,B01M7X0GAR |
| 552740 | B003S1UNZA,B00UGPRPE8 |
| 552741 | B003FZM7VC,B01M5IQG86 |
| 552742 | B003FZM7VC,B01M8PNC9K |
| 552743 | B004IN21VQ,B01GWC2H2I |
| 552744 | B004IN21VQ,B01M5IMMH3 |
| 552745 | B004IN21VQ,B01I063206 |
| 552746 | B004IN21VQ,B00UGQG8E  |
| 552747 | B003S1UNZU,B01GWCCK7S |
| 552748 | B003S1UNZU,B003XEXY8  |
| 552749 | B003S1UNZU,B076F6VMV7 |
| 552750 | B003S1UNZU,B000NKRJLU |
| 552751 | B003S1UNZU,B01GWC0I0M |
| 552752 | B003S1UNZU,B0002KPI28 |
| 552753 | B003S1UNZU,0780630459 |
| 552754 | B003S1UNZU,B006YJ6Z3C |
| 552755 | B003S1UNZU,B01M4QNK5  |
| 552756 | B003S1UNZU,B000NKNSUQ |
| 552757 | B003S1UNZU,B000NKPGRY |
| 552758 | B003S1UNZU,B0000ITUDO |
| 552759 | B003S1UNZU,B000NKKNC  |
| 552760 |                       |

## 4. 节点, 关系列表 -> 图数据库

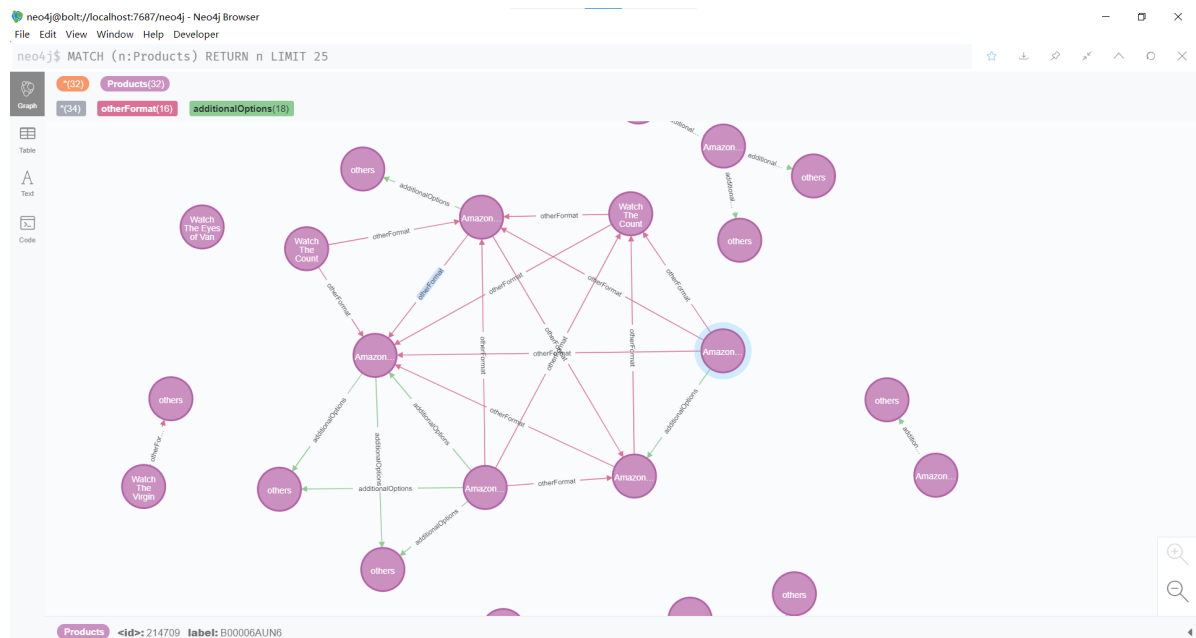
- 得到节点和关系之后, 即可导入neo4j数据库进行计算

```

1 // 导入节点
2 :auto USING PERIODIC COMMIT
3 LOAD CSV WITH HEADERS FROM "file:///node.csv" AS line
4 CREATE (p:Products{label:line.label,title:line.title})
5
6 // 导入otherFormat关系
7 :auto USING PERIODIC COMMIT
8 LOAD CSV WITH HEADERS FROM "file:///otherFormat.csv" AS line
9 MATCH (from:Products{label:line.label1}), (to:Products{label:line.label2})
10 MERGE (from)-[:otherFormat]-(to)
11
12 // 导入additionalOptions关系
13 :auto USING PERIODIC COMMIT
14 LOAD CSV WITH HEADERS FROM "file:///additionalOptions.csv" AS line
15 MATCH (from:Products{label:line.label1}), (to:Products{label:line.label2})
16 MERGE (from)-[:additionalOptions]-(to)

```

- 导入成功后, 即可在Neo4j Browser中看到数据库中存储的节点和节点之间的关系



- 最后调用Neo4j的图算法库

```
1 // 创建一个图
2 CALL gds.graph.create.cypher(
3 'mygraph',
4 'MATCH (n:Products) RETURN id(n) AS id',
5 'MATCH (a:Products)-->(b:Products) RETURN id(a) AS source, id(b) AS
target'
6)
7 YIELD graphName, nodeCount, relationshipCount, createMillis;
8
9 // 计算wcc(弱连通分量)
10 CALL gds.wcc.stats('mygraph')
11 YIELD componentCount
```

A screenshot of the Neo4j web interface showing a table view of the query results. The table has a single column named 'componentCount' and one row with the value '117514'. The interface includes a top menu bar, a left sidebar with icons for Table, Text, and Code, and a bottom status bar showing the query: 'CALL gds.wcc.stats('mygraph') YIELD componentCount'. The status bar also displays a message: 'Started streaming 1 records in less than 1 ms and completed after 177 ms.'

| componentCount |
|----------------|
| 117514         |

- 即可得到最终结果, 说明在25万条商品信息中一共出现了 117,514 部不同的电影

## 四、总结

本项目中, 我们通过开源数据集与Amazon网站爬取的商品数据, 完整地进行了数据的获取、数据的ETL和预处理。

