构建知识图谱与商品推荐

环境准备

在pycharm项目中安装graphframes

```
pip install graphframes
```

在windows环境使用，需要获取graphframes连接spark的jar包，可从graphframes官方寻找适合版本下载

```
https://spark-packages.org/package/graphframes/graphframes
```

Version: 0.8.2-spark3.1-s_2.12 (1cd7ab | zip | jar) / Date: 2021-10-17 / License: Apache-2.0 / Scala version: 2.12

Version: 0.8.2-spark3.2-s_2.12 (1cd7ab | zip | jar) / Date: 2021-10-17 / License: Apache-2.0 / Scala version: 2.12

Version: 0.8.1-spark2.4-s_2.11 (b3b97c | zip | jar) / Date: 2020-09-06 / License: Apache-2.0 / Scala version: 2.11

Version: 0.8.1-spark2.4-s_2.12 (b3b97c | zip | jar) / Date: 2020-09-06 / License: Apache-2.0 / Scala version: 2.12

Version: 0.8.1-spark3.0-s_2.12 (b3b97c | zip | jar) / Date: 2020-09-06 / License: Apache-2.0 / Scala version: 2.12

Version: 0.8.0-spark3.0-s_2.12 (8a81b8 | zip | jar) / Date: 2020-03-21 / License: Apache-2.0 / Scala version: 2.12

Version: 0.8.0-spark2.4-s_2.11 (8a81b8 | zip | jar) / Date: 2020-03-21 / License: Apache-2.0 / Scala version: 2.11

Version: 0.7.0-spark2.4-s_2.11 (f9e13a | zip | jar) / Date: 2019-01-08 / License: Apache-2.0 / Scala version: 2.11

Version: 0.7.0-spark2.3-s_2.11 (f9e13a | zip | jar) / Date: 2019-01-08 / License: Apache-2.0 / Scala version: 2.11

Version: 0.6.0-spark2.3-s_2.11 (3e6cfe | zip | jar) / Date: 2018-08-21 / License: Apache-2.0 / Scala version: 2.11

Version: 0.6.0-spark2.2-s_2.11 (3e6cfe | zip | jar) / Date: 2018-08-21 / License: Apache-2.0 / Scala version: 2.11

spark2.4.5环境使用graphframes-0.8.0-spark2.4-s_2.11.jar，下载后，将其放在项目中与代码同级位置

创建SparkSession入口时，指定jar包文件

```
from pyspark.sql import SparkSession
from graphframes import GraphFrame
ss = SparkSession.builder \
    .config("spark.jars", "graphframes-0.8.0-spark2.4-s_2.11.jar") \
    .getOrCreate()
```

一、实体分析

实体是实际存在的事物，在图中作为节点，对于用户行为数据，可识别出三种实体：

- 用户
- 商品
- 商品类型

根据实体构建点表，表中包含id、type字段**（id不可重复）**，其中type的值可以为'user'、'item'、'category'，示例如下：

id	type
10001082	user
10001083	user
100002303	item
3368	category
.....	.....

提示：

1、获取三种实体的无重复id表

2、每个表各添加一个type列，值为一个常量，例如商品表：

```
from pyspark.sql import functions as F
v_item = df_item.select('item_id').distinct()  #取id去重
v_item = v_item.withColumn('type',F.lit('item'))  #增加常量列
v_item = v_item.withColumnRenamed('item_id','id')  #重命名
```

3、将处理后的三个实体表用union合并

二、关系分析

任意实体间可能存在一定关系，且关系可包含属性，对于用户行为数据，可列举出以下关系：

- 用户——[点击]—>商品
- 用户——[收藏]—>商品
- 用户——[加购]—>商品
- 用户——[购买]—>商品
- 商品——[属于]—>类型

设计边表，其中用户的操作类型（1点击2收藏3加购4购买）作为关系名称，另外可增加操作日期date、操作次数count等属性，商品无这些属性可设为0，示例：

src	dst	relationship	date	count
10001082	100002303	1	2014-12-08	1
10001083	100002303	4	2014-12-08	3
100002303	3368	belong	0	0
.....	.....	.....	.....	.....

三、图构建

1、将上述点表、边表构建为图

2、节点初筛选，以度为标准，排除度极少的点，保留较重要的实体，构建新图

3、pagerank分析重要的节点

4、图可视化，若新图的点仍较多，可随机抽取一部分进行可视化

四、推荐查询

基于知识图谱进行商品推荐的目标是找到行为相似用户的喜好，主要使用find路径匹配再通过filter筛选

find具体使用方法：

```
find('模式')
#模式语法：
#(v1)-[e1]->(v2) 表示从v1点到v2点有一条e1边
#如果路径有两条边以上，以分号;分隔
#其中v1、e1、v2为自命名的形式参数
#查询结果可通过filter进一步筛选
```

示例：

通过find对用户10001082进行商品推荐

1、构建子图（只保留购买行为）

```
#设e是所有关系表
e1 = e.filter(e['relationship']=='4')
g1 = GraphFrame(v,e1)
```

2、查询购买相同商品的其他用户

```
#u1代表目标用户，u2代表其他用户，v代表共同商品
df1 = g1.find('(u1)-[e1]->(v);(u2)-[e2]->(v)')
df1.show()
```

```
+----------------+------------------+----------------+----------------+--------------------+
|              u1|                e1|               v|              u2|                  e2|
+----------------+------------------+----------------+----------------+--------------------+
|[103650067, user]|[103650067, 26037...|[260377643, item]|[103650067, user]|[103650067, 26037...|
| [32033746, user]|[32033746, 260377...|[260377643, item]|[103650067, user]|[103650067, 26037...|
|[103650067, user]|[103650067, 29572...|[295728407, item]|[103650067, user]|[103650067, 29572...|
|[106531595, user]|[106531595, 18066...|[180660092, item]|[106531595, user]|[106531595, 18066...|
|[107443646, user]|[107443646, 90074...| [90074271, item]|[107443646, user]|[107443646, 90074...|
```

3、筛选特定用户和购买行为，得到相似用户列表

```
df1 = df1.filter("u1.id = '10001082' AND e1.relationship = 4 AND e2.relationship
= 4").select('u2')
#收集用户id
users = []
for i in df1.collect():
    id = i['u2']['id']
    users.append(id)
print(users)
```

```
['64884837', '64884837', '10001082']
```

4、查询相似用户购买过的商品

```
df1 = e1.filter(F.col('src').isin(users)).select('dst')
#收集商品id
items = []
for i in df1.collect():
    id = i['dst']
    items.append(id)
print(items)
```

```
['53616768', '141312432', '220586551', '275221686', '225943470', '244315759', '275221686', '27522
```

5、检查待推荐商品是否被目标用户购买过

```
target = []
for i in items:
    data = e1.filter(e1['src']=='10001082' and e1['dst']==i)
    if data.count() == 0:
        target.append(i)
print(target)
```

最终推荐结果：

```
['225943470', '244315759', '295080192']
```