300行python代码从零开始构建基于知识图谱的电影问答系统3-实验环境和实验数据准备

原创

闰土不用叉 于 2019-05-06 20:56:39 发布 14269 收藏 66

分类专栏： 没事就搞搞系统来玩玩 没事就搞搞玩玩 文章标签： 知识图谱 neo4j 智能问答系统

本文链接：https://blog.csdn.net/xyz1584172808/article/details/89891248

没事就搞搞系统来玩玩 同时被 2 个专栏收录

5 篇文章 6 订阅

订阅专栏

没事就搞搞玩玩

5 篇文章 3 订阅

订阅专栏

> 貌似很久没有写了，这段时间一直在忙着准备复试，就有点耽误了，好吧，今天继续写。~~你们的魔鬼又来啦~~（什么鬼

在上一篇中，我对整个系统的业务逻辑啰里啰唆的梳理了一遍，如果你被我绕晕了，那也没关系，因为不用看上面那篇也能继续往下走，当你自己理清楚他的逻辑的时候，你就会有一种踏破铁鞋无觅处，柳暗花明又一村的感觉，好吧下面言归正传。

这一篇主要介绍实验的准备工作，也就是为后续工作铺平道路，主要包含实验环境和实验数据准备两部分，那么接下来就依次介绍这两个。

1、实验环境的搭建

（1）python环境的安装

本教程是基于python实现的，所以最基本的python环境要有，我的python版本是3.6，建议直接下载对应版本的anaconda来安装python，python环境搭建的具体细节这里就不做过多的介绍了，网上有很多，也可以直接参考anaconda官网。
接下来是安装依赖库，我把本项目涉及到的依赖库打包好了，可以直接使用下面的命令来安装：

```
pip install -r requirements.txt
```

哦，对了，这个项目基于webpy库来搭建的，上面的库里面已经包含了，但是这里安装可以会出错，那就等其他库装好了再来装他。

2019年5月10日09:03:18更新：webpy安装方式：

web.py 0.39 is the latest released version of web.py. You can install it by running:

```
pip install web.py
```

The above version only supports Python 2. If you looking for Python 3 support, try the experimental version.

```
pip install web.py==0.40-dev1
```

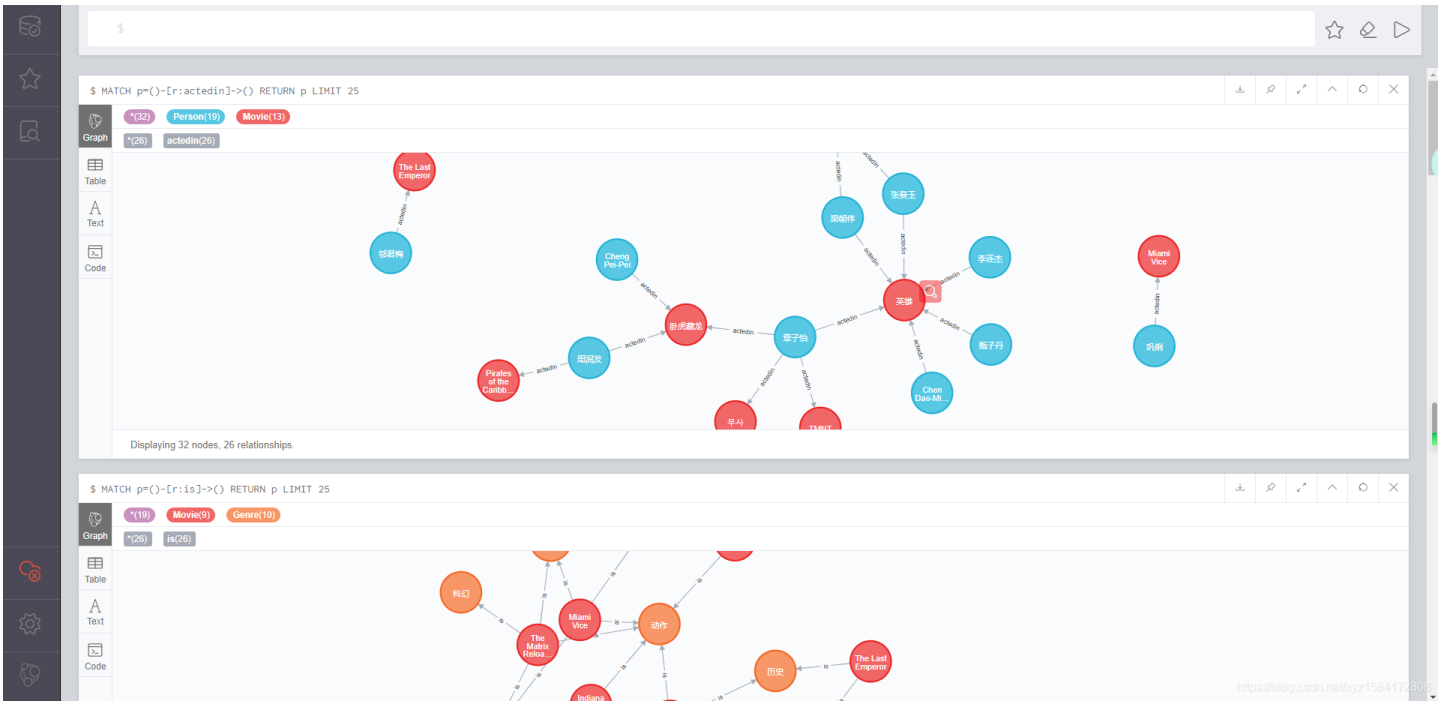
Or to get the latest development version from git:

```
git clone git://github.com/webpy/webpy.git
ln -s `pwd`/webpy/web .
```


通过上述步骤，那么基本的环境就搭建好了，这好比修一座房子，四面墙砌好了。

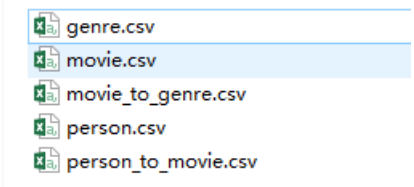
（2）知识图谱基本介绍

关于知识图谱，我自己也没研究得多深，所以请大家移步机器之心科普性的文章→ 这是一份通俗易懂的知识图谱技术与应用指南，读了这篇文章，相信你对知识图谱有了一个初步的印象，其实质就是利用三元组来表示实体的一些信息，而关于这些信息的存储，一种是基于RDF的存储；另一种是基于图数据库的存储。而本项目采用的是图数据库存储，主要是图数据库较简单，还直观，要是你搞了半天，一个像样一点的东西都没搞出来，你还有继续搞下去的信心吗？于是我就选择了图数据库neo4j，关于这个数据库的安装，请参照关于ubuntu下neo4j的安装与使用不管你是在win下安装，还是linux下安装都是可以的，因为这些坑我都踩过了，另外，我这里给出来的链接都是我之前实验时，成功了的教程，如果我再重复写一遍，貌似也没有多少意思，倒不如直接把他们的链接给出来，同时也我对这些文章的感谢。好了，到了现在，python的开发环境有了，知识图谱的存储环境有了，就差数据了，所谓万事俱备只欠东风。



2、实验数据

数据来源于IMDB数据库，这是一个关于电影演员、电影、电视节目、电视明星和电影制作的在线数据库。我们需要哪些数据呢？想想我们的目标是啥，我们要查询某部电影的信息，比如电影的评分，上映时间、大体讲了啥内容等，也可能查询某位演员的个人信息，演了那些电影等，好了，这里就出现了两个实体，电影，人物，然后人物和电影之间有着直接的关系：**[act]**，即某人出演了某部电影，于是可以用这个关系连链接演员和电影，此外，图数据库的好处在这里就凸显出来了，除了这种实体与实体的关系外，图数据库还可以链接实体和他的属性，所以，为了说明这种情况，单独构造了电影的类型，他是电影的属性，电影和他之间的关系是**is**。那么理下关系，主要有电影和人物这两个实体，他们之间的关系是出演，比如李连杰出演了《卧虎藏龙》，除此之外，实体还有自己的属性，特别的，对电影的类型这一属性单独处理。数据的格式如下：
全部数据文件：



人物实体数据文件：

	pid	birth	death	name	biography	birthplace
2	695	1937/3/16	1999/4/14	乔宏		Shanghai, China
3	1336	1963/4/26	\N	李连杰	李连杰（J	Beijing, China
4	1337	1962/6/27	\N	梁朝伟		Hong Kong

5	1338	1964/9/20	\N	张曼玉	Hong Kong
6	1339	1979/2/9	\N	章子怡	Beijing - China
7	1340	1955/4/26	\N	Chen Dao-Ming	Tianjin, China
8	1341	1963/7/27	\N	甄子丹	Yuexiu District, Guangzhou, China
9	1619	1955/5/18	\N	周润发	周润发，广Lamma Island, Hong Kong
10	1624	1946/12/4	\N	Cheng Pei-Pei	Shanghai, China
11	1633	\N	\N	鲍德熹	\N
12	10885	1938	\N	曾江	Hong Kong
13	11401	1946/9/22	\N	吴宇森	Guangzhou, China
14	12466	1964/7/20	\N	张耀扬	Hong Kong

人物和电影之间的关系链接：

	A	D	C
1	pid	mid	
2	163441	13	
3	240171	24	
4	1336	79	
5	1337	79	
6	1338	79	
7	1339	79	
8	1340	79	
9	1341	79	
10	643	82	
11	695	87	

这里的pid指的是personid，mid指的是movieid，是不是和传统的关系型数据库很像。

前面对数据的情况进行了介绍，那么怎么获取这样的数据呢？一种方法是爬去imdb上的数据，按照上面的说明来处理，另一种当然是直接下载我这处理好的数据啦，***然后把数据csv文件放入neo4j安装目录下的import目录下即可，***
我的路径是：

```
E:\neo4j-community-3.5.3-windows\neo4j-community-3.5.3\import
```

数据地址：链接：https://pan.baidu.com/s/1HgjZFQ7q4V_8EzzjNmMwwQ
提取码：7qv1
然后你们参照着这个来找吧。放进去后打开图数据库，在图数据库中依次执行下面的代码（*2019年7月27日更新*）：

```
//导入节点 电影类型  == 注意类型转换
LOAD CSV WITH HEADERS  FROM "file:///genre.csv" AS line
MERGE (p:Genre{gid:toInteger(line.gid),name:line.gname})


//导入节点 演员信息
LOAD CSV WITH HEADERS FROM 'file:///person.csv' AS line
MERGE (p:Person { pid:toInteger(line.pid),birth:line.birth,
death:line.death,name:line.name,
biography:line.biography,
birthplace:line.birthplace})


// 导入节点 电影信息
LOAD CSV WITH HEADERS  FROM "file:///movie.csv" AS line
MERGE (p:Movie{mid:toInteger(line.mid),title:line.title,introduction:line.introduction,
rating:toFloat(line.rating),releasedate:line.releasedate})


// 导入关系 actedin  电影是谁参演的 1对多
LOAD CSV WITH HEADERS FROM "file:///person_to_movie.csv" AS line
match (from:Person{pid:toInteger(line.pid)}),(to:Movie{mid:toInteger(line.mid)})
merge (from)-[r:actedin{pid:toInteger(line.pid),mid:toInteger(line.mid)}]->(to)

//导入关系  电影是什么类型 == 1对多
LOAD CSV WITH HEADERS FROM "file:///movie_to_genre.csv" AS line
match (from:Movie{mid:toInteger(line.mid)}),(to:Genre{gid:toInteger(line.gid)})
merge (from)-[r:is{mid:toInteger(line.mid),gid:toInteger(line.gid)}]->(to)
```

上一篇中提到了问题分类的训练集怎么构造的，也提到了问题模板，github上也有相关数据，可以直接下载，其中A是问题分类的训练数据，B是问题模板。

好吧，今天就写到这里了，我感觉又是一大堆废话，才写没多久，哎，大家先讲究着看，如果实在看不下去了就去找其他教程吧。希望以后空闲时间多一点的时候，多一些干活，少一些废话，保质保量。