漫威宇宙知识图谱构建-项目报告

作者：郝家辉 22021160 黄亦非 22021165 叶大源 22021214 姚璐22021213

1 项目背景及内容

漫威宇宙是由漫威影业基于漫威漫画角色制作的一系列电影组成的架空世界和共同世界。该共同世界是由共同的元素、设定、表演和角色通过跨界作品所建立的，并且与其它漫画、电影与动画等系列同属一个官方认可的多元宇宙。

本项目尝试为漫威宇宙中出现的主要角色，构建一个知识图谱，帮助我们更深入地了解漫威宇宙。

项目内容包括数据采集、知识存储、知识抽取、知识计算、知识应用五大部分，下文依次介绍对应工作。

2 数据采集与处理

2.1 数据采集

维基百科的漫威角色页面中包含了大量角色信息及长段的文本介绍，文本中记录着每个人物的具体故事以及与其他实体之间的交互信息，如下图所示。

由于百科网页的结构比较复杂，我们从中手工提取了有效信息文本单句，并进行了实体名的统一规范化，得到的数据如下。

```
X教授

X教授是一位在漫威漫画中的出版物《X战警》中的虚构人物。
X教授和至交好友万磁王不同，他一心想让变种人与人类能和平共处，但为了防止人类的迫害和引导变种人同胞走向正途，他以自身的财力创立了X学院来收容思绪不够成熟或不怎么会控制自己力量的变种人青少年，并建立了变种人的超级英雄团队X战警。

X教授的人物形象是参考美国非裔人权运动领袖马丁·路德·金恩，而X教授之名则参考另一位非裔人权运动领袖麦尔坎·X。

能力Edit
X教授是漫威漫画世界中最强大的心灵能力者。无需通过身体接触便能以此阅读他人的思维、记忆。同时他能通过其能力剥夺并控制他人的思维同时操纵他人的行动。

只要透过好友万磁王为他建造、位于X学院地下的X战警基地中的“脑波强化机”，X教授就能连结上世界上任何一个人，甚至只要他愿意的话，他可以依此杀死任何人。
```

为了进行数据标注，我们需要进一步将上述材料整理为较短的单句，得到的数据如下。

```
X教授是一位在漫威漫画中的出版物《X战警》中的虚构人物。
X教授和至交好友万磁王不同，他一心想让变种人与人类能和平共处。
X教授为了防止人类的迫害和引导变种人同胞走向正途，他以自身的财力创立了X学院来收容思绪不够成熟或不怎么会控制自己力量的变种人青少年。
X教授建立了变种人的超级英雄团队X战警。
X教授的人物形象是参考美国非裔人权运动领袖马丁·路德·金恩。
X教授之名则参考另一位非裔人权运动领袖麦尔坎·X。
X教授是漫威漫画世界中最强大的心灵能力者。
X教授无需通过身体接触便能以此阅读他人的思维、记忆。
X教授能通过其能力剥夺并控制他人的思维同时操纵他人的行动。
好友万磁王为X教授建造了、位于X学院地下的X战警基地中的“脑波强化机”。
利用脑波强化机，X教授就能连结上世界上任何一个人，甚至只要他愿意的话，他可以依此杀死任何人。
X教授在X战警电影系列中出现，包括《X战警》、《X战警2》、《X战警：最后战役》、《X战警：金钢狼》、《X战警：第一战》、《金钢狼：武士之战》、《X战警：未来昔日》及《X战警：天启》。
```

```
13  在头四及第六部电影中，X教授由派崔克·史都华饰演。
14  在第一战及天启中则由詹姆斯·麦艾维饰演年轻时的X教授。两名演员亦在未来昔日中一同饰演X教授。
```

2.2 数据标注

接着，我们借助精灵标注助手工具对提取出的文本进行实体关系标注。在标注过程中不断完善整个图谱的schema。最终确定包含13种实体类型，包括角色、作品、物种、机构、团队、装备、能力、演员、原型、职业、称号、计划、地点；25种关系类型，包括爱上、参与、持有、出现、创立、从事、代号、对抗、发现、雇佣、后代、来自、领导、配偶、朋友、去过、师从、饰演、收养、属于、兄弟姐妹、影射、拥有、制造、共生。

为了保证在这一过程中，我们小组成员并行进展时不出现过多的标注冲突与错误，我们建立了共享文档分享实体和关系信息。

实体

实体名	说明	举例	备注
角色	人物角色	钢铁侠	
作品	漫画或者电影的名字	《X战警》	含书名号
物种	角色或者生物的物种	人类、灭种人	
机构	国家部门等类似	国土安全局、X学院	
团队	角色组成的小队	复仇者联盟、X战警	
装备	角色的武器、设备等	雷神之锤、粉状强化机、无限手套	
能力	超能力描述、技能等	心灵控制、瞬间移动	
演员	演员名称	查理斯·扎	
原型	角色存在现实世界的原型	马丁路德金	
职业	角色的职业	教授、企业家	
称号	角色的称号或者代号	“宇宙最危险的女人” 钢铁侠 墨镜	不含引号
计划	xx计划	“重生计划”	不含引号
地点	美洲市、美地区、美国家	纽约、瓦拉达	

关系

关系名	关系格式	举例	备注
爱上	爱上 Arg1:角色,Arg2:角色		
参与	参与 Arg1:角色,Arg2:计划	张三加入了重生计划	
持有	持有 Arg1:角色,Arg2:装备	雷神和灭霸都拥有无限手套	
出现	出现 Arg1:角色,Arg2:作品 出现 Arg1:团队,Arg2:作品	死侍出现在电影《X战警》中	
创立	创立 Arg1:角色,Arg2:机构		
从事	从事 Arg1:角色,Arg2:职业	贾维斯是管家	
代号	代号 Arg1:角色,Arg2:称号	卡魔拉拥有称号“宇宙最危险的女人” 托尼史塔克代号钢铁侠	
对抗	对抗 Arg1:角色,Arg2:角色 对抗 Arg1:角色,Arg2:团队 对抗 Arg1:团队,Arg2:团队	唐发与尹戈是对抗/敌人关系	连一个方向就好
发现	发现 Arg1:角色,Arg2:角色	贝爵发现了卡魔拉	
雇佣	雇佣 Arg1:角色,Arg2:角色	史塔克雇佣了贾维斯	
后代	后代 Arg1:角色,Arg2:角色	勤乱是金刚猿的儿子	从孩子连长箭连
来自	来自 Arg1:角色,Arg2:地点	属前来自瓦拉达	
领导	领导 Arg1:角色,Arg2:团队 领导 Arg1:角色,Arg2:计划		
配偶	配偶 Arg1:角色,Arg2:角色	芭芭拉是盖眼的妻子	连一个方向就好
朋友	朋友 Arg1:角色,Arg2:角色	X教授和万磁王	连一个方向就好
去过	去过 Arg1:角色,Arg2:地点	灰的特克去华盛顿参加毕业旅行	
师从	师从 Arg1:角色,Arg2:角色		不一定拜师，师从教行
饰演	饰演 Arg1:演员,Arg2:角色	吴吉米由查理尔·外饰演	
收养	收养 Arg1:角色,Arg2:角色 收养 Arg1:团队,Arg2:角色	某人、团队收养某人	从长辈连孩子连
属于	属于 Arg1:角色,Arg2:机构 属于 Arg1:角色,Arg2:团队 属于 Arg1:团队,Arg2:团队 属于 Arg1:角色,Arg2:物种		前一个属于后一个
兄弟姐妹	兄弟姐妹 Arg1:角色,Arg2:角色		连一个方向就好
影射	影射 Arg1:角色,Arg2:原型	X教授原型是马丁路德金	
拥有	拥有 Arg1:角色,Arg2:能力 拥有 Arg1:团队,Arg2:能力		
制造	制造 Arg1:角色,Arg2:装备	万磁王为X教授制作了超级强化机	
共生	共生 Arg1:角色,Arg2:角色	毒液和人类共生	连一个方向就好

角色实体	文事别名	文事别名	文事别名
X教授			
埃德温·贾维斯			
埃里克·恩维格			
艾尔佛特·E·罗斯			
艾瑞克·乔尔奥格			
艾维·佛斯特	幽灵	ghost	
托尼·斯塔克	安东尼·爱德华·斯塔克	钢铁侠	托尼
安雅			
维彼比邻星			
奥创			
奥科耶			
比尔·佛斯特			
彼得·帕克			
布莱恩·布拉多克	美国队长		
布伦希尔德	女武神	瓦尔基里	
查克			
查理·赫斯廷	斯凯	莫波	神盾局的人
查理·维克	毒液	埃迪·布洛克	闪电·洛雷桑
多米诺			
菲尼·布雷泽	奇灵骑士		
菲兰多			
格丽西娅·哈代	黑豹		
格利普·科尔森	科尔森		
汪达·马克斯莫夫	绯红女巫		
塞蒂丝 哈罗德·霍顿			

标注完成后，精灵标注助手能够导出json格式。为方便后续任务的进行，需要对标注数据进行清洗与合并，消除标注产生的错误。

2.3 数据处理

2.3.1 数据合并

由于标注过程是由四个人并行完成，最终需要将数据合并成同一个json标注文件，这一过程会有标注实体的偏移量、编号发生顺延的情况出现，为此我们选择使用Python脚本自动化完成这一过程。

使用以下命令，确定脚本的两个待拼接的输入文件，然后确定输出文件，运行后即可得到拼接后的json标注文件。

```
1  python3 merge.py -p <preInputFile> -a <appendInputFile> -o <outputFile>
```

2.3.2 数据检查

为了确保合并过程中没有出现偏移量错误，我们使用Python脚本检查标注原文的偏移量位置文本是否与标注文本一致对应。

使用以下命令，即可检查标注文件的偏移量是否有错，如果有错误，会将标注附近源文本和标注内容及偏移量等信息提供给用户以供修正。

```
1  python3 check.py -i <inputFile>
```

2.3.3 数据修复

尽管采用了共享文档等措施避免标注冲突和错误，但仍然会有一小部分问题出现，例如英译名称不同等。为此，我们使用Python脚本来修订错误。

使用以下命令，即可修复标注文件的错误，对于出现冲突的标注文件，脚本会询问用户正确标注应该是什么，然后对源文本和标注以及偏移量等信息进行修正。

```
python3 repair.py -i <inputFile> -o <outputFile>
```

2.3.4 数据增强

在数据标注中，存在着很多的关系是无向的，但是为了提高标注速度，我们仅标注了单向，因此需要在这一过程中补充另一个方向的关系，保证其无向性。

使用以下命令即可完成数据增强过程。

```
python3 bio.py -i <inputFile>
```

2.4 数据转化

对数据集进行基本的处理后，我们得到了正确、完整的json标注数据。下面我们要为需要进行的知识存储、计算、推理和抽取等工作转化出对应格式的数据。

使用以下命令即可得到DeepKE所需的数据文件、用于知识存储的N-Triple文件。

```
python3 tran.py
```

3 知识存储

3.1 基于Apache Jena三元数据库的存储

3.1.1 Jena介绍

Jena是语义Web领域最为流行的开源Java框架和RDF数据库之一，它支持的三元组格式包括RDF/XML、Turtle、N-Triple和RDFa。此前我们已生成N-Triple格式的文件，可以直接用于导入数据。N-Triple格式文件使用一行数据表示主谓宾三元组，示例如下。

```
<http://kg.course/marvel/角色/X教授> <http://kg.course/marvel/relation/创立>
<http://kg.course/marvel/机构/X学院> .
```

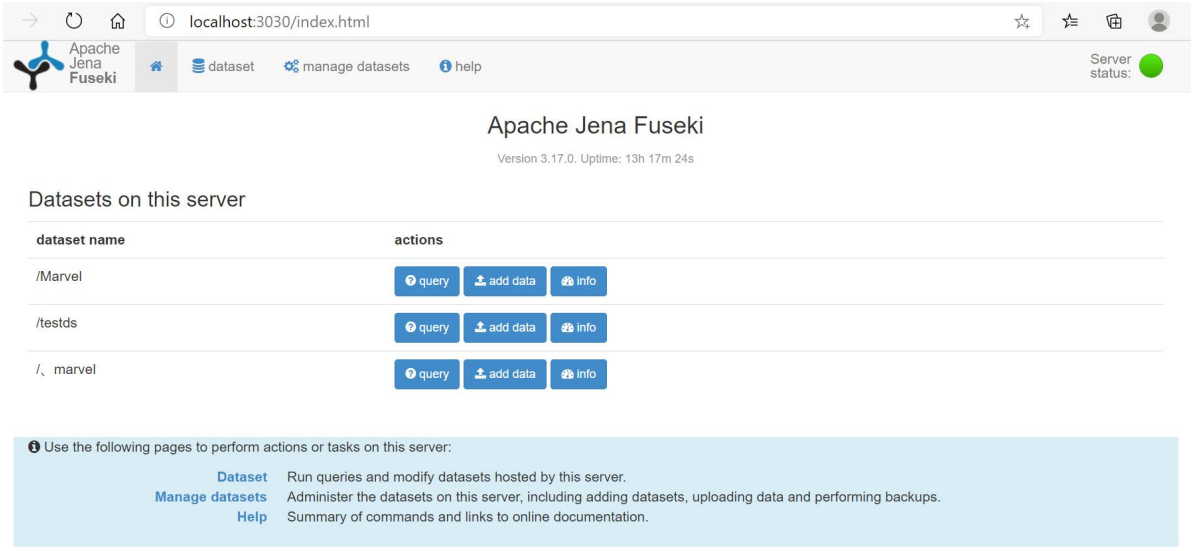
Fuseki是基于Jena的服务器，支持SPARQL语句的查询。在Jena官网可以下载最新版本的Fuseki，解压至本地后通过命令行启动服务。为了支持数据的上传，可以提前创建文件夹store再修改配置文件shiro.ini将其中一行配置改为/$/** = anon，最后再启动服务。具体操作可参考文档。

3.1.2 启动Fuseki服务与上传数据

在已经完成解压、建立文件夹、修改默认配置文件的操作后，通过命令行进入解压的文件夹中，然后启动服务。启动时在命令行有如下日志显示

```
C:\Users\Reeve\Desktop\C++\apache-jena-fuseki-3.17.0\apache-jena-fuseki-3.17.0>fuseki-server --loc=store --update /testds
09:47:19 INFO  Server          :: Running in read-only mode for /testds
09:47:19 INFO  Server          :: Apache Jena Fuseki 3.17.0
09:47:20 INFO  Config          :: FUSEKI_HOME=C:\Users\Reeve\Desktop\C++\apache-jena-fuseki-3.17.0\apache-jena-fuseki-3.17.0\.
09:47:20 INFO  Config          :: FUSEKI_BASE=C:\Users\Reeve\Desktop\C++\apache-jena-fuseki-3.17.0\apache-jena-fuseki-3.17.0\run
09:47:20 INFO  Config          :: Shiro file: file://C:\Users\Reeve\Desktop\C++\apache-jena-fuseki-3.17.0\apache-jena-fuseki-3.17.0\run\shiro.ini
09:47:22 INFO  Config          :: Template file: templates/config-tdb-dir
09:47:23 INFO  Server          :: Database: TDB1 dataset: location=store
09:47:23 INFO  Server          :: Path = /testds
09:47:23 INFO  Server          :: System
09:47:23 INFO  Server          ::   Memory: 1.0 GiB
09:47:23 INFO  Server          ::   Java:   1.8.0_211
09:47:23 INFO  Server          ::   OS:     Windows 10 10.0 amd64
09:47:23 INFO  Server          ::   PID:    9776
09:47:23 INFO  Server          :: Started 2021/01/16 09:47:23 CST on port 3030
```

启动时会自动配置本机的3030端口，之后可以借助浏览器与Fuseki服务交互，进而使用Jena。在浏览器打开3030端口可见类似如下界面，其中右上角指示灯为绿色说明数据连接正常，中间为创建的数据集，数据集可以通过数据集管理按键点进带有创建功能的页面，创建数据集时可以选择该数据集的数据是否需要做持久化操作。



在上图的actions部分点击“add data"按钮可以在数据集上传已提前生成的N-Triple文件。数据上传完成后在数据集（dataset）页面点击"edit"按键，可以查看Jena根据上传的三元组信息自动生成的另一种以主语为根谓语为枝宾语为叶的树状三元组组织形式，并且可以在该形式下直接进行数据修改和保存。Jena自动生成的三元组组织示例如下。

```
<http://kg.course/marvel/角色/幻视>
        a       <http://kg.course/marvel/角色> ;
        <http://kg.course/marvel/relation/name>
                <http://kg.course/marvel/角色/幻视> ;
        <http://kg.course/marvel/relation/出现>
                <http://kg.course/marvel/作品/《复仇者联盟3：无限战争》> ,
<http://kg.course/marvel/作品/《美国队长3：内战》> , <http://kg.course/marvel/作品/《复仇者联盟4》> , <http://kg.course/marvel/作品/《复仇者联盟2：奥创纪元》> ;
        <http://kg.course/marvel/relation/对抗>
                <http://kg.course/marvel/角色/奥创> ;
        <http://kg.course/marvel/relation/属于>
                <http://kg.course/marvel/团队/复仇者联盟> , <http://kg.course/marvel/团队/西岸复仇者> ;
        <http://kg.course/marvel/relation/配偶>
                <http://kg.course/marvel/角色/汪达·马克希莫夫> .
```

3.1.3 SPARQL查询示例

SPARQL是一种为RDF开发的查询语言和数据获取协议，被Jena和Fuseki所支持。下列几张图片是一些简单的查询尝试，包含SPARQL代码与返回结果。下列展示的查询分别查询彼得·帕克（蜘蛛侠）的装备，鹰眼（克林特·巴顿）所爱上的角色，九头蛇的对手，电影《黑豹》当中出现的角色。

- 查询彼得·帕克的装备

```
PREFIX relation:<http://kg.course/marvel/relation/>
PREFIX zp:<http://kg.course/marvel/作品/>
PREFIX ch:<http://kg.course/marvel/称号/>
PREFIX js:<http://kg.course/marvel/角色/>
PREFIX marvel:<http://kg.course/marvel/>
SELECT DISTINCT ?object
WHERE {
  js:彼得·帕克 relation:持有 ?object
}
LIMIT 10
```

IERY RESULTS

Table | Raw Response

owing 1 to 2 of 2 entries

object
marvel:装备/蛛网发射器
marvel:装备/蜘蛛侠战衣

- 查询鹰眼（克林特·巴顿）喜欢的角色

```
PREFIX relation:<http://kg.course/marvel/relation/>
PREFIX zp:<http://kg.course/marvel/作品/>
PREFIX ch:<http://kg.course/marvel/称号/>
PREFIX marvel:<http://kg.course/marvel/>
SELECT DISTINCT ?object
WHERE {
  ?subject relation:代号 ch:鹰眼 .
  ?subject relation:爱上 ?object
}
LIMIT 10
```

ERY RESULTS

Table | Raw Response

wing 1 to 7 of 7 entries

object
marvel:角色/汪达·马克希莫夫
marvel:角色/娜塔莉娅·罗曼诺夫
marvel:角色/芭芭拉·莫尔斯
marvel:角色/月光石
marvel:角色/珍妮特·范·达因
marvel:角色/娜塔莉亚·罗曼诺娃
marvel:角色/杰西卡·德鲁

- 查询九头蛇反派组织的对手（因数量过多，增加了查询的数量限制）

```
PREFIX relation:<http://kg.course/marvel/relation/>
PREFIX marvel:<http://kg.course/marvel/>
PREFIX team:<http://kg.course/marvel/团队/>
SELECT DISTINCT ?subject
WHERE {
  ?subject relation:对抗 team:九头蛇
}
ORDER BY ?subject
LIMIT 10
```

在该查询中，返回的结果包括角色类型也包括团队类型，若要只查询角色类型，可以在查询语句当中增加约束条件

```
  ?subject rdf:type marvel:角色
```

- 查询《黑豹》当中出现的角色

```
PREFIX relation:<http://kg.course/marvel/relation/>
PREFIX zp:<http://kg.course/marvel/作品/>
PREFIX marvel:<http://kg.course/marvel/>
SELECT DISTINCT ?角色
WHERE {
  ?角色 relation:出现 zp:《黑豹》

}
ORDER BY ?角色
LIMIT 10
```

上列截图中，结果以表格形式呈现，事实上Fuseki支持以JSON、XML、CSV、TSV格式返回搜索结果。以查询彼得·帕克的装备为例，JSON格式的返回结果为


```
{ "head": {
    "vars": [ "object" ]
  } ,
  "results": {
    "bindings": [
      {
        "object": { "type": "uri" , "value": "http://kg.course/marvel/装备/蛛网发射器" }
      } ,
      {
        "object": { "type": "uri" , "value": "http://kg.course/marvel/装备/蜘蛛侠战衣" }
      }
    ]
  }
}
```


3.2 基于Neo4j的存储设计

3.2.1 Neo4j简介

Neo4j(Network Exploration and Optimization 4 Java)是由Neo4j公司开发的图数据库管理系统。Neo4j的开发者将其描述为一个符合ACID标准的事务性数据库，它具有原生的图存储和处理功能，Neo4j有GPL3授权的开源”社区版"，其在线备份和高可用性扩展等功能则以闭源商业许可的方式进行授权。

Neo4j是用Java实现的，通过事务性的HTTP终端或二进制的"bolt"协议，可以利用使用Cypher查询语言的其他语言编写的软件中对Neo4j数据库进行访问。

3.2.2 Neosemantics

简介

Neosemantics是一个Neo4j中的一个插件，它基于Neo4j原生插件APOC，利用Neosemantics，我们可以在Neo4j中使用RDF。RDF是一个W3C标准的数据交换模型。这实际上意味着Neosemantics可以实现：

- 以无损的方式将RDF数据存储在Neo4j中，也就是导入后的RDF在导出后不会丢失任何一个triple。
- 可以根据需要将Neo4j中的属性图数据导出为RDF供其他流程使用。
- 在Neo4j的图上进行模型映射和推断。

安装

在Neo4j中安装Neosemantics插件有两种方法以供选择：

- 从源代码编译
- 使用预编译二进制`jar`包安装

为了保证可迁移性，我们使用第二种方法进行Neosemantics的安装。具体步骤如下：

1. 从GitHub Neosemantics主页的发布区下载适合本机环境的预编译`jar`包，本项目使用版本为`4.1.0.1`。
2. 新建Neo4j工程，并在插件区安装APOC插件。
3. 将`neosemantics-4.1.0.1.jar`拷贝至`<NEO_HOME>/plugins`目录下。
4. 在`<NEO_HOME>/conf/neo4j.conf`文件中添加以下内容，标记Neosemantics插件生效。

```
dbms.unmanaged_extension_classes=n10s.endpoint=/rdf
```

重启服务器后，在命令区输入以下命令检查安装是否成功。

```
call dbms.procedures()
```

查询到`n10s`命名空间下的方法即表明安装成功。

```
neo4j$ call dbms.procedures()
```

	name	signature
		NODE?)"
353	"n10s.inference.nodesInCategory"	"n10s.inference.nodesInCategory(category :: NODE?, params = {} :: MAP?) :: (node :: NODE?)"
354	"n10s.inference.nodesLabelled"	"n10s.inference.nodesLabelled(label :: STRING?, params = {} :: MAP?) :: (node :: NODE?)"
355	"n10s.mapping.add"	"n10s.mapping.add(elementUri :: STRING?, graphElementName :: STRING?) :: (schemaNs :: STRING?, schemaPrefix :: STRING?, schemaElement :: STRING?, elemName :: STRING?)"
356	"n10s.mapping.drop"	"n10s.mapping.drop(graphElementName :: STRING?) :: (output :: STRING?)"
357	"n10s.mapping.dropAll"	"n10s.mapping.dropAll(namespace :: STRING?) :: (output :: STRING?)"

Started streaming 387 records after 3 ms and completed after 22 ms.

数据准备

在Neo4j中导入RDF的主要方法是`n10s.rdf.import.fetch`。它将URI返回的三组数据导入并持久化到Neo4j中，这个URI可以指向RDF文件（本地或远程）或动态生成RDF的服务。

我们使用的是`N-Triple`格式的RDF文件，即前文中处理得到的`data/marvel.nt`。在导入RDF文件前，首先要对图进行初始化，然后再使用上述方法导入。

```
CALL n10s.graphconfig.init();
CREATE CONSTRAINT n10s_unique_uri ON (r:Resource)
ASSERT r.uri IS UNIQUE;
CALL n10s.rdf.import.fetch("file:///D:\\data\\marvel.nt", "N-Triples");
```

输入命令后，我们成功导入了3849个triple。

```
neo4j$ CALL n10s.rdf.import.fetch("https://box.zjuqsc.com/~84544425", "N-Triples");
```

	terminationStatus	triplesLoaded	triplesParsed	namespaces
1	"OK"	3849	3849	{ "ns1": "http://kg.course/marvel/relation/", "ns0": "http://kg.course/marvel/" }

Started streaming 1 records after 8 ms and completed after 1200 ms.

3.2.3 Cypher查询

简介

Cypher是一种声明式图查询语言，可以在属性图中进行表达式的高效数据查询。Cypher主要是Andrés Taylor在2011年为Neo4j公司工作时的发明。Cypher最初打算与图数据库Neo4j一起使用，但在2015年10月通过openCypher项目独立开放。

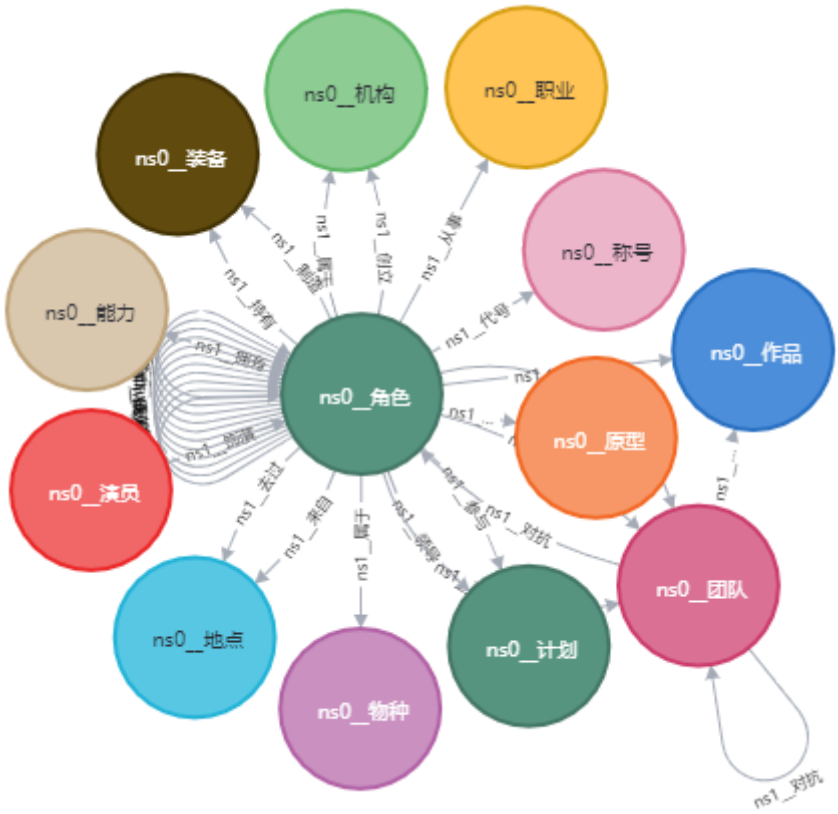
该语言在设计时考虑到了SQL（关系型数据库模型的标准查询语言）的强大和能力，但Cypher是基于建立在图论概念上的数据库的组件和需求。在图模型中，数据被结构化为节点（顶点）和关系（边），以关注数据中的实体如何相互连接和关联。

Schema可视化查询

输入以下命令即可可视化查询Schema的结构

```
call db.schema.visualization();
```

查询结果如下图所示，图中的节点代表知识图谱中的实体，节点之间的边代表知识图谱中的关系。

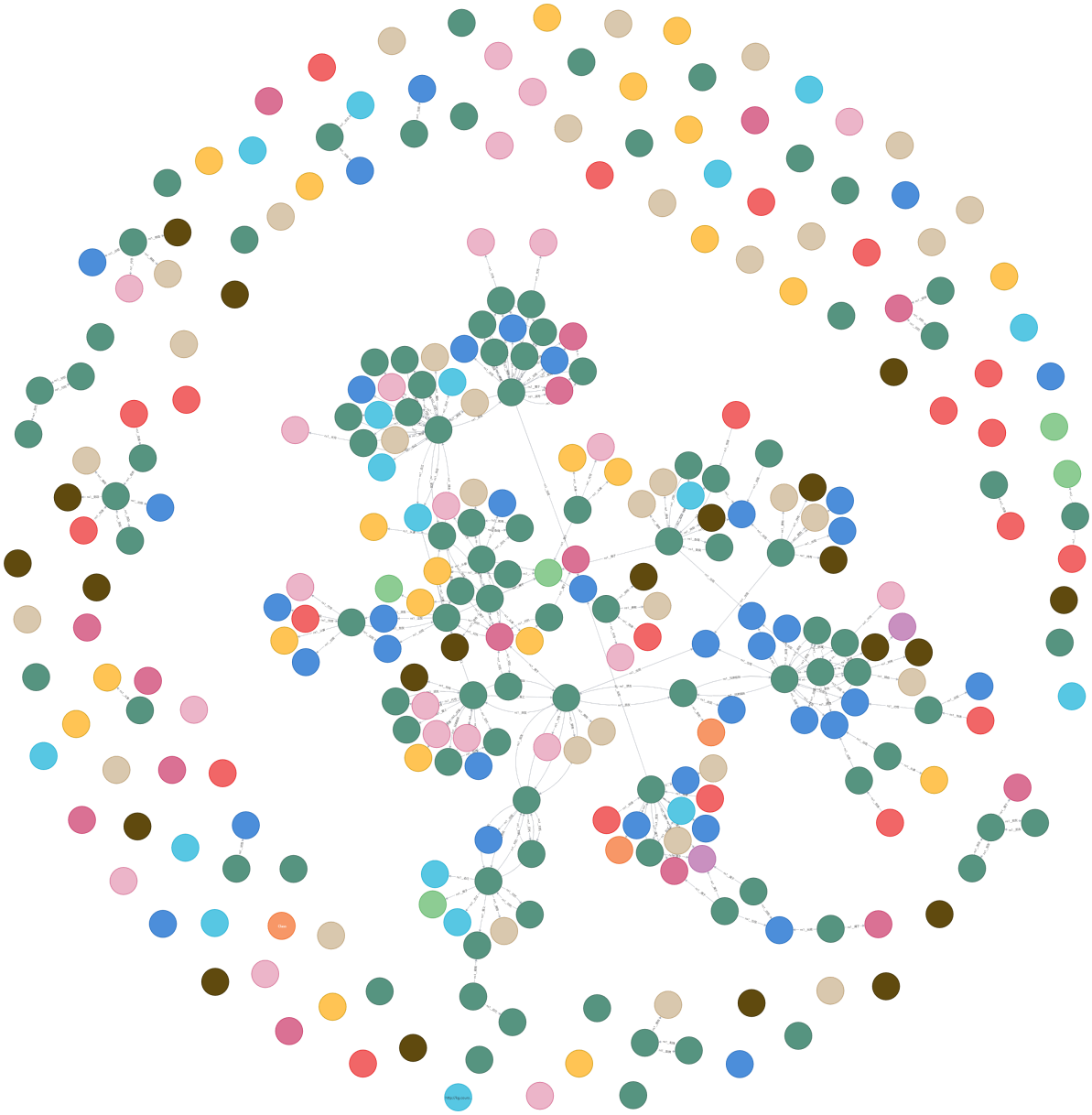


知识图谱全局查询

输入以下命令即可得到整个知识图谱的全貌

```
call db.schema.visualization();
```

查询结果如下图所示



利用Cypher查询进行简单的知识挖掘

Cypher可以支持各种各样的查询语句，可以利用其高度定制化的查询方式进行简单的知识挖掘。

- 家族查询

  我们通过以下命令查询`角色`类型中名字包含`斯塔克`的节点，这潜在的意义即为查询`斯塔克`家族的情况。

```
MATCH (n:ns0__角色)
WHERE n.uri CONTAINS '斯塔克' RETURN n;
```

  查询结果如下图所示，我们查询到了`458-托尼·斯塔克`和他的父母`485-霍华德·斯塔克`和`451-玛丽亚·斯塔克`，他们三人的关系也十分明了。

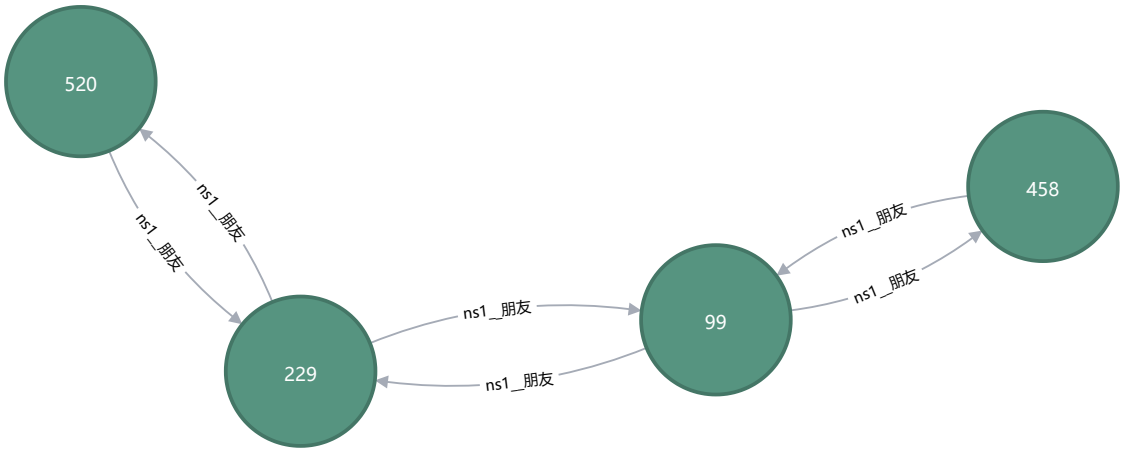
- 六度分隔理论

  可以使用Cypher查询任意两个人的关系，检验六度分隔理论是否正确。

  > 六度分隔理论认为世界上任何互不相识的两人，只需要很少的中间人（6人）就能够建立起联系。

```
MATCH p=shortestPath( (n1)-[*]-(n2) )
WHERE n1.uri CONTAINS '托尼·斯塔克' and n2.uri CONTAINS '凤凰'
and all( x in nodes(p)
    where 'ns0__角色' IN LABELS(x))
RETURN p;
```

查询结果如下图所示，我们发现经过两个中间人：`229-布莱恩·布拉多克`和`99-彼得·帕克`就可以建立起`520-凤凰`到`458-托尼·斯塔克`之间的联系。



4 知识抽取

DeepKE是陈华均老师课题组开发的，基于Pytorch深度学习框架的一款高效的关系抽取框架，在这部分我们使用DeepKE对我们之前得到的数据集进行中文的关系抽取的实践和测试。

4.1 标注转换和统计

在该小节，我们需要完成的是以下几部分的内容：

1. 将数据转换为DeepKE接收的csv的格式
2. 随机的将数据集划分为7:2:1的比例，作为训练集、测试集和验证集。

在我们的项目当中训练集、测试集和验证集的个数分别为：736,210,106。

输出的文件在`deepke/data/marvel/out`当中，原始的csv数据保存在`deepke/data/origin/`文件夹当中。

```
|--- data
     |--- origin
          |--- relation.csv
          |--- train.csv
          |--- valid.csv
          |--- test.csv
     |--- marvel
          |--- out #output folder
```

4.2 训练关系抽取模型

在训练过程当中，我们使用了DeepKE当中提供的`cnn, rnn, capsure, transformer, gcn, LM`等模型进行了相关的训练和测试验证。训练的参数都使用的项目当中的默认参数。epoch数我们修改为50，`num_relations`在我们的项目当中，算上None type总共有`32`种。下面是训练各种模型所使用的命令：

```
# train cnn model
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out
num_relations=32 epoch=50 model=cnn gpu_id=4

# train rnn model
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out
num_relations=32 epoch=50 model=rnn gpu_id=4
```

```
# train capsure model
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=capsure gpu_id=4

# train transformer model
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=transformer gpu_id=4

# train gcn model
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=gcn gpu_id=4

# train BERT with hidden layer=1
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=lm lm_file=/private/huangyifei/pretrain/bert_base_chinese gpu_id=5 num_hidden_layers=1

# train BERT with hidden layer=2
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=lm lm_file=/private/huangyifei/pretrain/bert_base_chinese gpu_id=5 num_hidden_layers=2

# train BERT with hidden layer=3
python main.py show_plot=False data_path=data/origin out_path=data/marvel/out num_relations=32 epoch=50 model=lm lm_file=/private/huangyifei/pretrain/bert_base_chinese gpu_id=5 num_hidden_layers=3
```

4.3 训练结果

	CNN	RNN	Capsure	Transformer	GCN	BERT layer=1	BERT layer=2	BERT layer=3
Valid	81.13	74.53	72.64	78.30	18.87	82.08	**83.02**	80.19
Test	72.86	70.48	64.76	72.86	17.14	78.57	**79.52**	78.10

从训练的结果上我们可以看到，基于BERT这种预训练语言模型然后在关系抽取任务上进行微调的方法相对来说取得了最佳的效果，这也证明在小数据集上预训练语言模型抽取的特征还是有着很强的泛化能力的。而GCN的结果相对来说是最差的。

5 知识计算

“图计算”是以“图论”为基础的对现实世界的一种“图”结构的抽象表达，以及在这种数据结构上的计算模式。它是研究客观世界当中的任何事物和事物之间的关系，对其进行完整的刻划、计算和分析的一门技术。

本章节中我们基于Neo4j进行图计算。

5.1 角色网络分析

- 角色数量

  通过以下命令即可查询出知识图谱中的角色数。

  ```
  MATCH (c:ns0__角色) RETURN count(c);
  ```

  查询结果如下，共有221位角色出现在了知识图谱中。

```
neo4j$ MATCH (c:ns0__角色) RETURN count(c)
```

Table

"count(c)"
221

Text

- 角色关系概要

  通过以下命令即可统计每个角色过的其它角色的数目。

```
MATCH (c:`ns0__角色`)-[]->(:`ns0__角色`)
WITH c, count(*) AS num
RETURN min(num) AS min, max(num) AS max, avg(num) AS avg_characters, stdev(num)
AS stdev
```

  查询结果如下，共有221位角色出现在了知识图谱中。每个角色平均解决近3名角色，最多达到了26名，最少仅为1名，标准差为3.79。

```
neo4j$ MATCH (c:`ns0__角色`)-[]→(:`ns0__角色`) WITH
```

Table

"min"	"max"	"avg_characters"	"stdev"
1	26	2.7168674698795163	3.7912920729389046

Text

- 角色在图中的关系直径

  可以通过以下命令查询到角色在图中的关系直径。

```
MATCH (a:`ns0__角色`), (b:`ns0__角色`) WHERE id(a) > id(b)
MATCH p=shortestPath((a)-[*]-(b))
RETURN length(p) AS len, [x IN nodes(p) | split(x.uri,
'http://kg.course/marvel')[-1]] AS path
ORDER BY len DESC LIMIT 4
```

  在图中角色的最大关系直径达到了8，也就是说即便再延申关系，也不会长于这一直径。

```
neo4j$ MATCH (a:`ns0__角色`), (b:`ns0__角色`) WHERE id(a) > id(b) MATCH p=shortestPath((a)-[*]-(b)) RETURN length(p) AS len, [x IN nodes(p) | split(x.u
```

Table

	len	path
1	8	["/角色/亚特兰蒂斯", "/角色/纳摩", "/物种/变种人", "/角色/韦德·威尔逊", "/角色/死亡女神", "/角色/萨诺斯", "/角色/洛基·劳菲森", "/作品/《雷神》", "/角色/埃里克·塞维格"]
2	8	["/角色/里德·理查德", "/装备/可以伸出能量爪的手套", "/角色/喇叭", "/角色/韦德·威尔逊", "/角色/彼得·帕克", "/作品/《复仇者联盟3：无限战争》", "/角色/洛基·劳菲森", "/作品/《雷神》", "/角色/简·福斯特"]
3	8	["/角色/亚特兰蒂斯", "/角色/纳摩", "/物种/变种人", "/角色/韦德·威尔逊", "/角色/死亡女神", "/角色/萨诺斯", "/角色/洛基·劳菲森", "/作品/《雷神》", "/角色/简·福斯特"]
4	8	["/角色/里德·理查德", "/装备/可以伸出能量爪的手套", "/角色/喇叭", "/角色/韦德·威尔逊", "/角色/死亡女神", "/角色/萨诺斯", "/角色/洛基·劳菲森", "/作品/《雷神》", "/角色/埃里克·塞维格"]

Text

Warn

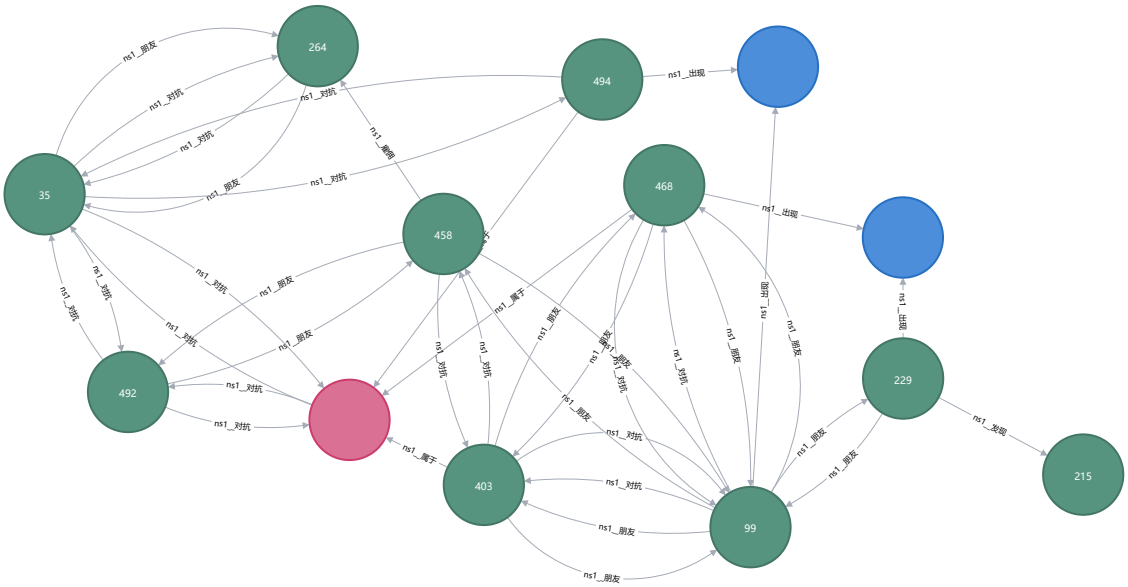
Code

- 角色间的最短路径

  可以通过以下命令查询到角色之间的所有最短关系路径。我们查询 35-奥创 到 215-亚瑟王 之间的所有最短路径

```
MATCH (n1:`ns0__角色`), (n2:`ns0__角色`)
WHERE n1.uri CONTAINS '奥创' and n2.uri CONTAINS '亚瑟王' and id(n2) > id(n1)
MATCH p=allShortestPaths((n1)-[*]-(n2))
RETURN p
```

查询结果如下图所示，路径中的节点数最小为6。



- 角色间的关键节点

  角色之间的最短路径可能存在多条，如果一个节点位于所有的最短路径上，那么称之为关键节点。

```
MATCH (a:`ns0__角色`), (b:`ns0__角色`) WHERE id(a) > id(b)
MATCH p=allShortestPaths((a)-[*]-(b)) WITH collect(p) AS paths, a, b
MATCH (c:`ns0__角色`) WHERE all(x IN paths WHERE c IN nodes(x)) AND NOT c IN
[a,b]
RETURN a.uri, b.uri, c.uri AS PivotalNode SKIP 490 LIMIT 10
```

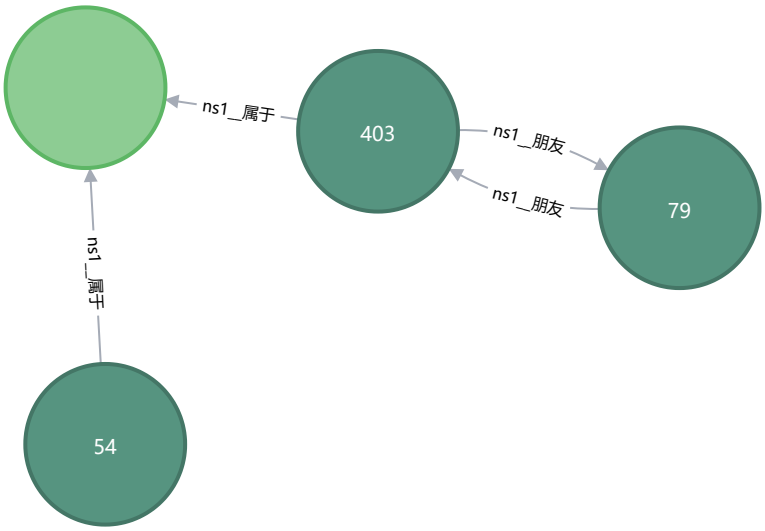
查询结果如下图所示，知识图谱中所有的关键节点都被列出。

"a.uri"	"b.uri"	"PivotalNode"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/愚弑"	"http://kg.course/marvel/角色/史蒂文·罗杰斯"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/威廉·贝克"	"http://kg.course/marvel/角色/彼得·帕克"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/威廉·贝克"	"http://kg.course/marvel/角色/史蒂文·罗杰斯"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/贾盈"	"http://kg.course/marvel/角色/黛西·约翰逊"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/贾盈"	"http://kg.course/marvel/角色/史蒂文·罗杰斯"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/简·福斯特"	"http://kg.course/marvel/角色/托尔·奥丁森"
"http://kg.course/marvel/角色/猎鹰"	"http://kg.course/marvel/角色/雪绪"	"http://kg.course/marvel/角色/韦德·威尔逊"
"http://kg.course/marvel/角色/亨利·皮姆"	"http://kg.course/marvel/角色/威廉·贝克"	"http://kg.course/marvel/角色/彼得·帕克"
"http://kg.course/marvel/角色/亨利·皮姆"	"http://kg.course/marvel/角色/贾盈"	"http://kg.course/marvel/角色/黛西·约翰逊"
"http://kg.course/marvel/角色/亨利·皮姆"	"http://kg.course/marvel/角色/雪绪"	"http://kg.course/marvel/角色/韦德·威尔逊"

我们以第一条，从`猎鹰`到`愚弑`的关键节点为`史蒂夫·罗杰斯`进行查询，验证是否正确。

```
MATCH (n1:`ns0__角色`), (n2:`ns0__角色`)
WHERE n1.uri CONTAINS '猎鹰' and n2.uri CONTAINS '愚弑' and id(n1) <> id(n2)
MATCH p=shortestPath((n1)-[*]-(n2))
RETURN p
```

查询结果如下图所示，`403-史蒂夫罗杰斯`是关键节点。



5.2 节点中心度分析

在图论和网络分析中，中心性指标可以确定图中最重要的顶点。应用包括识别社会网络中最有影响力的人、互联网或城市网络中的关键基础设施节点以及疾病的超级传播者。中心性概念最早是在社会网络分析中发展起来的，许多用于衡量中心性的术语反映了它们的社会学起源。

- 建立临时图

  为了执行后面的图计算的方便，我们建立一个角色到角色之间的临时图。

```
CALL gds.graph.drop('my-cypher-graph');
CALL gds.graph.create.cypher(
    'my-cypher-graph',
    'MATCH (n:ns0__角色) RETURN id(n) AS id',
    'MATCH (a:ns0__角色)-[]->(b:ns0__角色) RETURN id(a) AS source, id(b) AS target'
);
```

- PageRank方法

  PageRank算法根据传入关系的数量和对应源节点的重要性，衡量图内每个节点的重要性。其基本假设大致是，一个页面的重要性只与链接到它的页面一样。

  使用以下命令就可以计算出各个角色的PageRank分数。

```
CALL gds.pageRank.stream('my-cypher-graph')
YIELD nodeId, score
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1] AS name, score
ORDER BY score DESC, name ASC
```

  查询结果如下图所示，蜘蛛侠、英国队长、死侍、钢铁侠以及美国队长等热门角色的得分名列前茅。

neo4j$ CALL gds.pageRank.stream('myG…

Table | Text | Code

	name	score
1	"彼得·帕克"	4.691619491437451
2	"布莱恩·布拉多克"	3.718674402893521
3	"韦德·威尔逊"	3.6197532097576186
4	"托尼·斯塔克"	3.505732988100499
5	"史蒂文·罗杰斯"	2.9215510501991955
6	"彰弘"	2.2415858157211916

- ArticleRank方法

  ArticleRank是PageRank算法的一个变种，它衡量的是节点的转折性影响力或连接性。

  使用以下命令就可以计算出各个角色的ArticleRank分数。

```
CALL gds.alpha.articleRank.stream('my-cypher-graph')
YIELD nodeId, score
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1]
AS name, score
ORDER BY score DESC, name ASC
```

  查询结果如下图所示，与PageRank算法略有不同。

neo4j$ CALL gds.alpha.articleRank.stre…

Table | Text | Code

	name	score
1	"彼得·帕克"	1.1627532511511107
2	"韦德·威尔逊"	0.9174474063177909
3	"托尼·斯塔克"	0.8510548552615036
4	"布莱恩·布拉多克"	0.8111963889106817
5	"史蒂文·罗杰斯"	0.7381281572750824
6	"史考特·朗恩"	0.6133583410610756

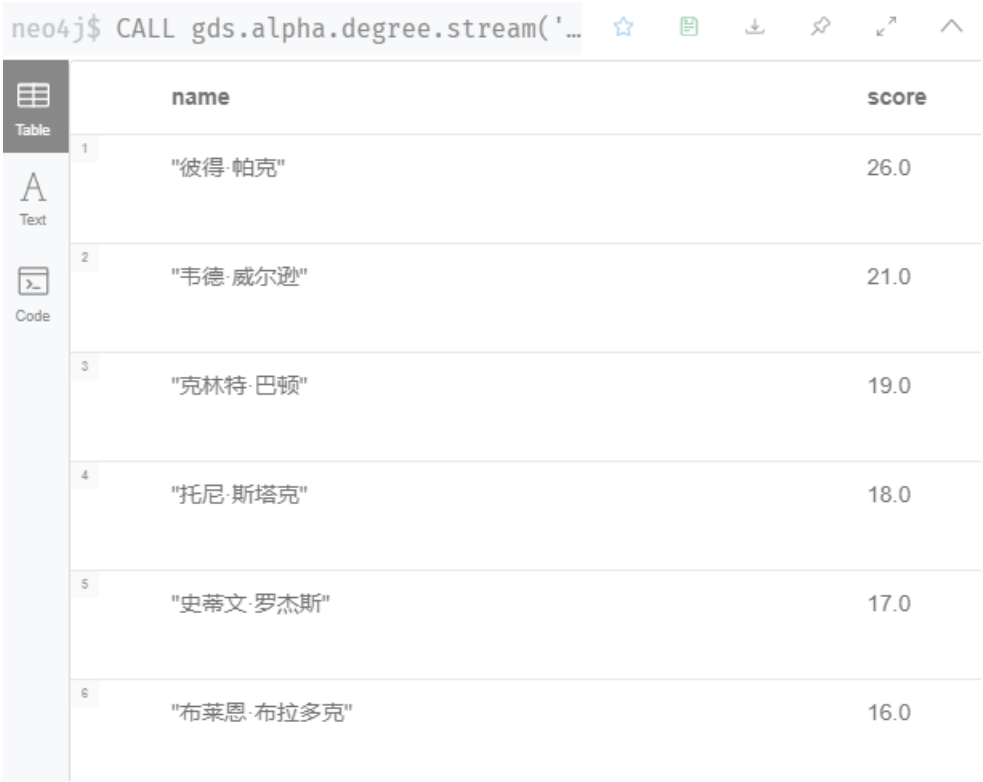
- 度中心性

度中心率衡量一个节点的传入和传出关系的数量，度中心性算法可以帮助我们找到图中的热门节点。

使用以下命令就可以计算出各个角色的度中心性。

```
CALL gds.alpha.degree.stream('my-cypher-graph')
YIELD nodeId, score
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1]
AS name, score
ORDER BY score DESC, name ASC
```

查询结果如下图所示，可以发现在漫威宇宙中，蜘蛛侠——彼得·帕克可以说是交际花，其度中心性极高。之后的为钢铁侠——托尼·斯塔克、死侍——韦德·威尔逊等人，都是漫威中的热门角色。



- 介数中心性

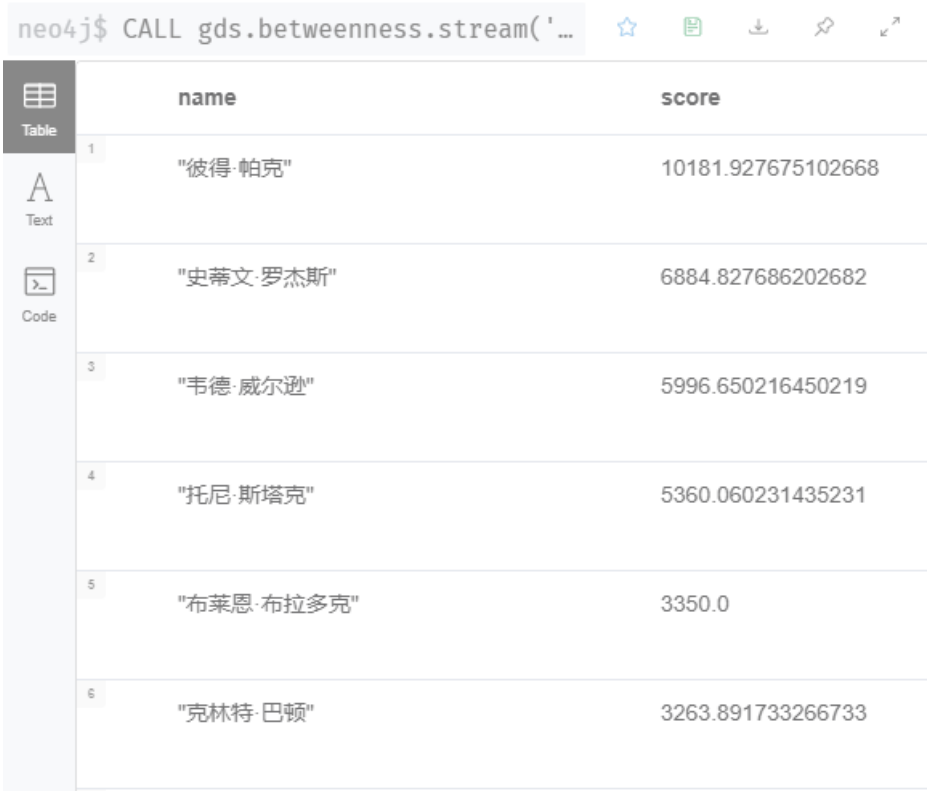
  介数中心性是一种检测一个节点对图中信息流的影响程度的方法。它经常被用来寻找作为图中一个部分到另一个部分的桥梁的节点。

  该算法计算图中所有节点对之间的非加权最短路径。根据通过该节点的最短路径数，每个节点都会得到一个分数。更频繁地位于其他节点之间的最短路径上的节点将有更高的中心性得分。

  在官方插件中安装`gds`插件后，即可使用以下命令就可以计算出各个角色的介数中心性。

```
CALL gds.betweenness.stream('my-cypher-graph')
YIELD nodeId, score
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1]
AS name, score
ORDER BY score DESC;
```

查询结果如下图所示，可以发现彼得·帕克也是介数中心性最高的节点，此后的排序与度中心性有所不同。



- 紧密度中心性

  紧密度中心性是一种检测能够在图中非常有效地传播信息的节点的方法。

  一个节点的紧密度中心性衡量它与所有其他节点的平均远度（反距离）。紧密度得分高的节点与所有其他节点的距离最短。

  在官方插件中安装`gds`插件后，即可使用以下命令就可以计算出各个角色的介数中心性。

```
CALL gds.alpha.closeness.stream('my-cypher-graph')
YIELD nodeId, centrality
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1]
AS name, centrality
ORDER BY centrality DESC;
```

  查询结果如下图所示，我们查看之前几位度中心性或介数中心性较高的角色的紧密度中心性可以发现，他们都处于中位水平。

neo4j$ CALL gds.alpha.closeness.strea…

Table | Text | Code

	name	centrality
18	"彼得·帕克"	0.5181518151815182
19	"史蒂文·罗杰斯"	0.5032051282051282
20	"亚特兰蒂斯"	0.5
21	"伦纳德·麦肯齐"	0.5
22	"托尼·斯塔克"	0.46865671641791046
23	"韦德·威尔逊"	0.45375722543352603

- 调和中心性

  调和中心性（又称值中心性）是紧密度中心性的一个变种，它的发明是为了解决原始公式在处理非连接图时的问题。与许多中心性算法一样，它起源于社会网络分析领域。

  使用以下命令就可以计算出各个角色的度中心性。

```
CALL gds.alpha.closeness.harmonic.stream('my-cypher-graph')
YIELD nodeId, centrality
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1]
AS name, centrality AS score
ORDER BY score DESC, name ASC
```

  查询结果如下图所示，可以发现和紧密度中心性差异不大。

neo4j$ CALL gds.alpha.closeness.harmon…

Table | Text | Code

	name	score
1	"彼得·帕克"	0.2825
2	"史蒂文·罗杰斯"	0.26946969696969697
3	"托尼·斯塔克"	0.25757575757575757
4	"韦德·威尔逊"	0.25545454545454543
5	"史考特·朗恩"	0.2292424242424242
6	"毒液"	0.22893939393939394

5.3 社区检测

社区检测算法用于评估节点组是如何聚类或分区的，以及它们加强或分离的趋势。Neo4j GDS库包括以下社区检测算法：

- Louvain
- Label Propagation
- Weakly Connected Components
- Triangle Count
- Local Clustering Coefficient

5.3.1 Louvain方法

Louvain方法是一种检测大型网络中社区的算法。它最大限度地提高了每个社区的模块化得分，其中模块化量化了节点分配到社区的质量。这意味着评估一个社区内的节点连接的密集程度，与它们在随机网络中的连接程度相比。

Louvain算法是一种分层聚类算法，它将社区递归合并为一个节点，并在浓缩图上执行模块化聚类。

执行以下命令即可执行Louvain算法。

```
CALL gds.louvain.stream('my-cypher-graph')
YIELD nodeId, communityId
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1] AS
name, communityId
ORDER BY communityId ASC
```

查询结果如下图所示，我们可以看出，英国队长相关的朋友和角色都被聚类到了70号社区。

类似的，以`菲尔·科尔森`为代表的神盾局雇员们都被聚类到了第144号社区。

5.3.2 Label Propagation方法

标签传播算法(Label Propagation algorithm，LPA)是一种在图中寻找社区的快速算法，它仅以网络结构为指导来检测这些社区，不需要预先定义目标函数或社区的先验信息。它仅以网络结构为指导来检测这些社区，不需要预先定义目标函数或社区的先验信息。

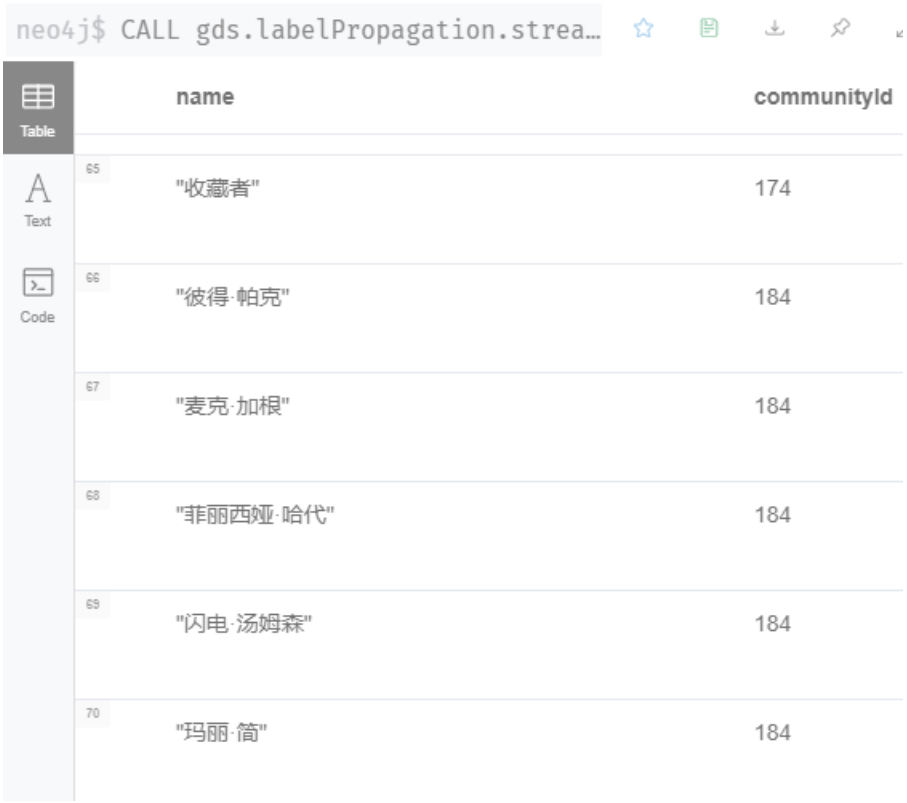
LPA的工作原理是在整个网络中传播标签，并基于这个标签传播的过程形成社区。该算法背后的直觉是，单个标签可以在密集连接的节点群中迅速成为主导，但将难以穿越连接稀疏的区域。标签会被困在一组密集连接的节点内部，当算法完成后，那些最终拥有相同标签的节点可以被认为是同一社区的一部分。

执行以下命令即可执行LPA算法。

```
CALL gds.labelPropagation.stream('my-cypher-graph')
YIELD nodeId, communityId
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1] AS
name, communityId
ORDER BY communityId ASC
```

查询结果如下图所示，这次我们检测了复仇者联盟这一社区。鹰眼、绯红女巫以及洛基等角色都被正确归入了一个社区内。

彼得·帕克和毒液等角色也被正确划分到了蜘蛛侠相关的社区内。



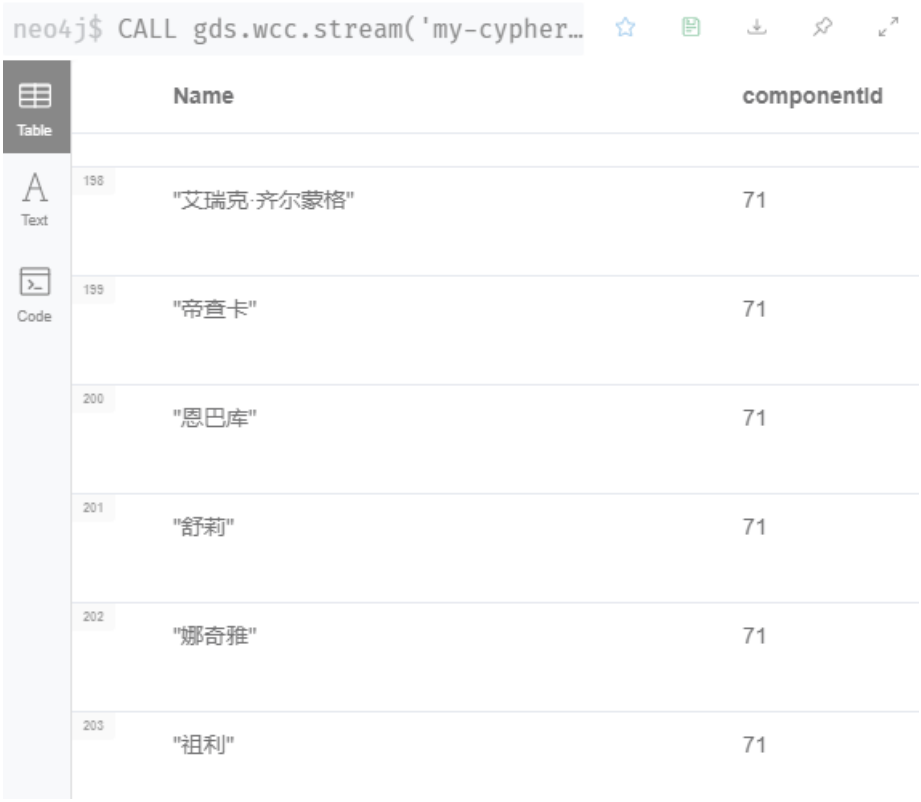
5.3.3 Weakly Connected Components方法

WCC算法在非定向图中寻找连接节点的集合，同一集合中的所有节点构成一个连接的分量。在分析的早期，WCC经常被用来理解图的结构。使用WCC来理解图结构，可以在确定的簇上独立运行其他算法。作为有向图的预处理步骤，它有助于快速识别不相连的群。

执行以下命令即可执行WCC算法。

```
CALL gds.wcc.stream('my-cypher-graph')
YIELD nodeId, componentId
RETURN split(gds.util.asNode(nodeId).uri, 'http://kg.course/marvel/角色/')[-1] AS
Name, componentId
ORDER BY componentId ASC
```

查询结果如下图所示，可以看到瓦坎达的各位部落首领都被归入了同一社区内。



6 知识应用

6.1 知识问答

为使用我们知识图谱实现一个简易的知识问答系统demo，我们使用了他人的开源项目并对我们的知识图谱作了适配。该工程首先实现分词，再对问句进行模板匹配，根据匹配结果找到相应的SPARQL语句模板进行填充并发送给Fuseki服务器。

该项目主要包括五个代码文件，文件主要信息如下表

代码文件名	主要用途
query_main.py	main函数，包含读入提问和对最终回答的包装与输出
word_tagging.py	实现自然语言到Word对象的方法。
question2sparql.py	对问句的处理：根据提问语句调用word_tagging进行分词，匹配question_temp中的模板得到SPARQL语句
questionn_temp.py	定义SPARQL模板和自然语言匹配规则，提供根据SPARQL模板封装SPARQL语句的函数
jena_sparql_endpoint.py	连接Fuseki服务器

我们的demo共支持九种提问，对应九种SPARQL语句模板如下。

```
#公共模板
PREFIX : <http://www.kgdemo.com#>
PREFIX relation:<http://kg.course/marvel/relation/>
PREFIX zp:<http://kg.course/marvel/作品/>
PREFIX ch:<http://kg.course/marvel/称号/>
PREFIX js:<http://kg.course/marvel/角色/>
PREFIX wz:<http://kg.course/marvel/物种/>
PREFIX jg:<http://kg.course/marvel/机构/>
PREFIX td:<http://kg.course/marvel/团队/>
PREFIX zb:<http://kg.course/marvel/装备/>
PREFIX nl:<http://kg.course/marvel/能力/>
PREFIX yy:<http://kg.course/marvel/演员/>
PREFIX yx:<http://kg.course/marvel/原型/>
PREFIX zy:<http://kg.course/marvel/职业/>
PREFIX jh:<http://kg.course/marvel/计划/>
PREFIX dd:<http://kg.course/marvel/地点/>
PREFIX marvel:<http://kg.course/marvel/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?s WHERE {
            <此处插入下表中的SPARQL语句模板>
}
```

提问类型	SPARQL模板
某角色有哪些朋友	?s relation:朋友 js:{person} .
某团队有那些成员	?s relation:属于 td:{team} . ?s rdf:type marvel:角色 .
某人属于哪些团队	js:{person} relation:属于 ?s . ?s rdf:type marvel:团队 .
某计划有谁参加	js:{person} relation:参与 ?s . ?s rdf:type marvel:计划 .
某作品出现了哪些角色、团队	{?s relation:出现 zp:{movie}.} UNION {?s relation:出现 zp:《{movie}》.}
某角色在哪些作品中出现	js:{person} relation:出现 ?s.
某角色去过哪些地方	{js:{person} relation:去过 ?s .} UNION {js:{person} relation:来自 ?s .}
某角色拥有哪些装备	js:{person} relation:持有 ?s . ?s rdf:type marvel:装备 .

为使我们的demo项目适配我们的知识图谱，主要进行了如下修改。首先是根据开源项目中的`question_temp.py`文件修改文本匹配模板和SPARQL模板形成`MyQuestionSet.py`文件，并替换原本的`question_temp.py`。然后是修改分词模块，我们试着把我们图谱中的实体名提取出来制作词典导入分词模块`word_tagging.py`，直接使用原项目中的工具，这种做法对于带有`·`符号的角色名效果不好；我们也尝试使用百度提供的分词工具`paddlepaddle-tiny`，该工具对于`猎鹰`等在生活中存在其他含义的名称在词性标注上效果不太好，如需进一步改进，或许可以在百度分词的基础上结合自制的词典做优化或采用其他的优化策略。最后在`jena_sparql_endpoint.py`修改与Fuseki服务器交互的url。

如需运行我们的智能问答demo，首先应根据3.1节内容启动Fuseki，然后使用python运行`query_main.py`文件即可。

问答演示：

```
>> 请输入问题：尼克·弗瑞的朋友有谁
Paddle enabled successfully......
彼得·帕克、菲尔·科尔森、娜塔莉娅·罗曼诺夫
########################################################################################################
>> 请输入问题：托尼·斯塔克有什么装备
Paddle enabled successfully......
汽车电池驱动的电磁铁、微型方舟反应堆、动力装甲、马克II号、马克V号装甲
########################################################################################################
>> 请输入问题：X教授出现在什么电影
Paddle enabled successfully......
《X战警》、《X战警2》、《X战警：最后战役》、《X战警：金钢狼》、《X战警：第一战》、《金钢狼：武士之战》、《X战警：未来昔日》、《X战警：天启》
########################################################################################################
>> 请输入问题：《X战警》有哪些英雄？
Paddle enabled successfully......
X教授、韦德·威尔逊、布莱恩·布拉多克、天启
########################################################################################################
```

6.2 知识图谱可视化

我们利用流行的可视化库D3来进行实体关系知识图谱的可视化。相关代码放在`visualization`目录下。

首先我们将之前的实体关系标注结果转换为可视化所需的数据格式。

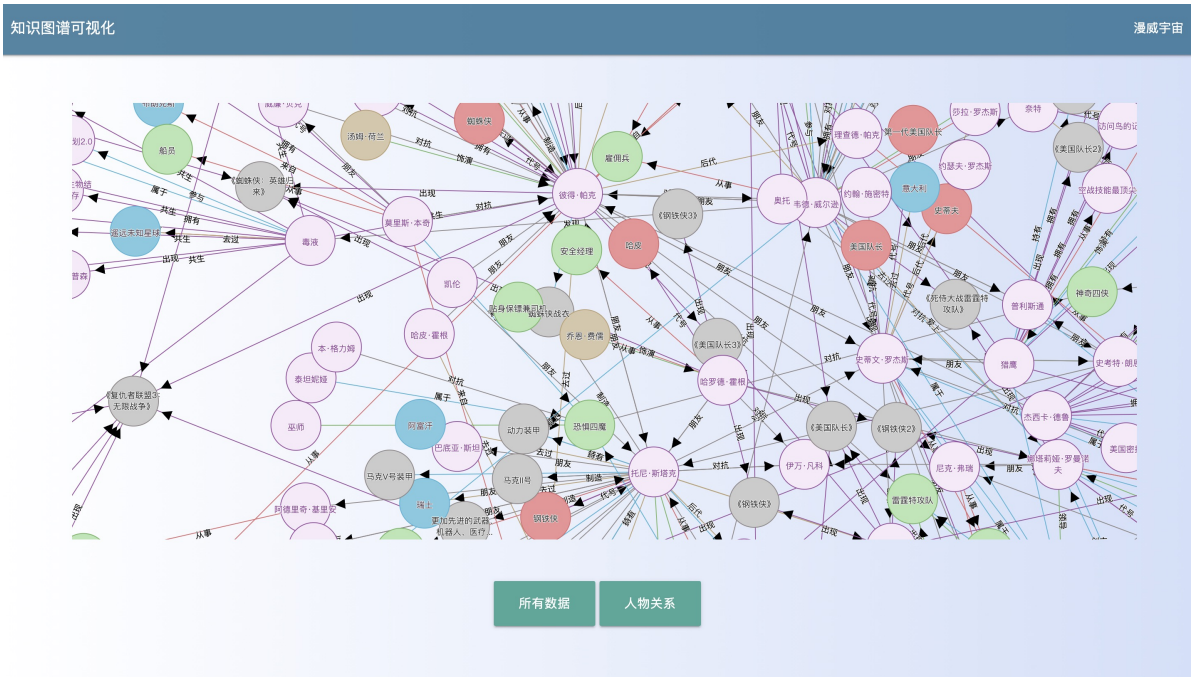
```
cd visualization
python anno2vizdata.py
python filter_vizdata.py
```

运行上述代码后会在`visualization/KGView`文件夹下生成`vizdata.json`和`vizdata_filtered.json`这两份数据文件，前者包含所有数据（包括武器、地点等），后者只包含人物关系和称号。

可视化网页存放于`visualization/KGView/index.html`中，可用chrome浏览器打开。

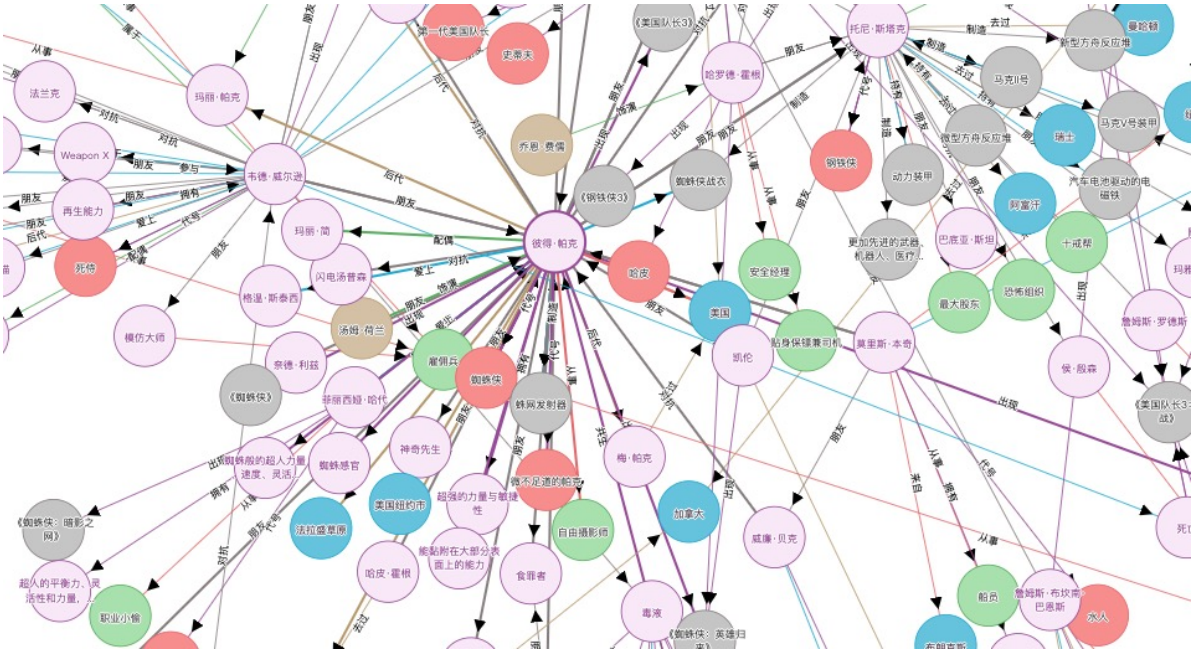
> 需要对浏览器先进行跨域配置，使其能够读取本地数据。可参考该教程进行配置。

打开后可视化界面如下图所示，默认打开的是**所有数据**模式。

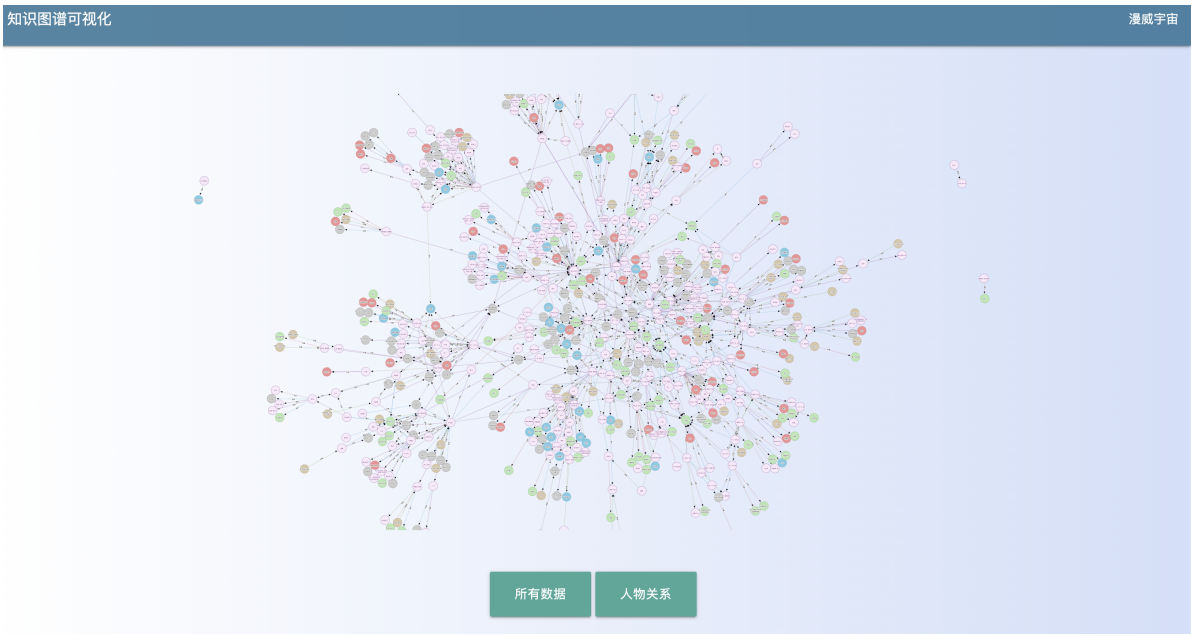


不同颜色的圆圈代表不同类型的实体，连线代表实体间的关系。每个节点可以被高亮选中、拖动，整体可以缩放，整个可视化是动态的。

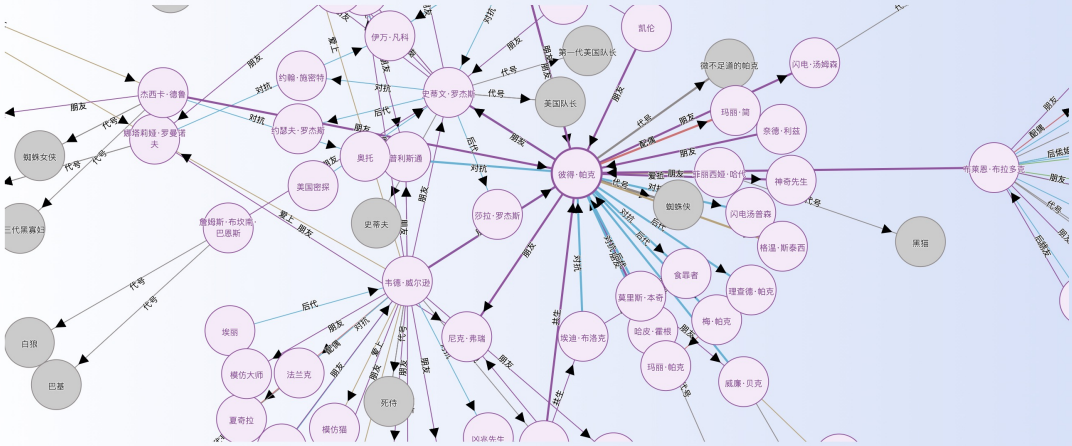
可以观察到主要人物拥有大量连线，意味着它与其他多个实体有着紧密联系。如下图中的”蜘蛛侠“彼得·帕克，”钢铁侠“托尼·斯塔克，”死侍“韦德·威尔逊。



缩放后可得到整体效果图如下。



点击下方按钮“人物关系”可以切换到更简洁的人物关系可视化，如下图所示。


杰西卡·德鲁
娜塔莉娅·罗曼诺夫
蜘蛛女侠
二代黑寡妇
白狼
巴基
约翰·施密特
约瑟夫·罗杰斯
美国密探
伊万·凡科
史蒂文·罗杰斯
奥托
普利斯通
史蒂夫
詹姆斯·布坎南·巴恩斯
韦德·威尔逊
邓拉·罗杰斯
埃丽
模仿大师
法兰克
夏奇拉
死侍
模仿猫
尼克·弗瑞
凯伦
第一代美国队长
美国队长
彼得·帕克
埃迪·布洛克
莫里斯·本奇
食蚁者
蜘蛛侠
哈皮·霍根
玛丽·帕克
梅·帕克
理查德·帕克
威廉·贝克
微不足道的帕克
闪电·汤姆森
玛丽·简
奈德·利兹
神奇先生
闪电汤普森
格温·斯泰西
黑猫
所有数据
人物关系