

明代历史知识图谱

「知识图谱导论」 课程作业

丁智 黄家名 李皓宇 孟特石

一、项目简介与分工

近年来，知识图谱（Knowledge Graph）逐渐走入我们的生活中。早在 2012 年 5 月 16 日，Google 为了提升搜索引擎返回的答案质量和用户查询的效率，发布了知识图谱。有了知识图谱作为辅助，搜索引擎能够洞察用户查询背后的语义信息，返回更为精准、结构化的信息，更大可能地满足用户的查询需求。Google 知识图谱的宣传语“Things not strings.” 给出了知识图谱的精髓，即，不要无意义的字符串，而是获取字符串背后隐含的对象或事物。所以我们建立知识图谱的目的就在于要建立起一个从数据到知识库中知识要素映射的过程，将知识库中的知识与问题或者数据加以关联，机器完全可以重现我们人的理解与解释过程。

前段时间，《觉醒年代》的播出唤醒了新时代青年的爱国热情。历史的车轮滚滚向前，新时代的我们，特别是工科生的我们，顺应浩浩荡荡时代潮流的同时，更要铭记历史，不忘先辈之初心，砥砺前行，为祖国未来的建设贡献自己的力量。



由此，我们小组决定将现代科技与历史相结合，构建明代历史人物关系知识图谱，铭记历史，不忘初心，砥砺前行。本项目实现了基于明代历史人物及事件的知识图谱，让使用者沉浸在纵横开阔的历史人文知识海洋中。我们的「明代历史知识图谱」项目的关系图谱展示了关于条目主体（如朱元璋）的所有关联性信息，使用者可以通过某个主体的关系图谱，一步一步探索未知的关系可能性。

本项目中小组成员与分工情况如下表所示。

小组成员	学号	主要工作
丁智	12121074	构建 MySQL 数据库以存储实体属性及实体间关系，设计并实现知识图谱实体及关系剪枝程序
黄家名	12121123	研究 neo4j 图数据库，从 mysql 数据库中读取数据并写入图数据库中，并对数据进行可视化展示
李皓宇	22121028	研究网站数据信息，通过编写优化后的 DFS 算法，实现将网站的信息爬取下来，并存入 Mysql 数据库中。
孟特石	22121104	调研包含目标数据的网站，确认可用数据来源，开发网站单条数据爬取的 API，并将其封装为库函数，供后续调用。

二、具体技术实现

本项目中的文件结构

2.1 图谱结构设计

本知识图谱包含 10 种类型的 939 个实体和 341 种类型的 1501 条关系及其属性。

其中，939 个实体包含的类型有：历史人物、历史事件、地点、战争、权利机构、作品、历史时期、法律等 9 个确定的类型，对于其他实体个数较少的类型全部归到“其他”这一类别。

关系的类型包括出生于、平定、父子、修订、出版物、朋友、删除、认知、属于、影响、画作、源自等，用以描述两个实体之间的关系，针对这一条知识，还有其对应的介绍，具体组织形式可见 2.2 部分的 mysql 数据表。

2.2 数据采集

本组调研了近 10 个网站，考虑到数据的可靠性与数据获取的难度，通过对比分析，最终选择“全历史”网站与“百度百科”这两个网站作为本次课程设计项目的数据来源。

在数据采集方面，我们使用了 Python 语言，基于 selenium+urllib3 框架，开发爬虫项目，实现爬取网页信息的任务，并使用 BeautifulSoup+正则表达式的方式对网页信息进行解析，剔除网页中的冗余信息，以提取出我们需要的核心数据，完成我们初步的数据获取任务。

在数据清洗方面，为了避免出现与主题相关性过低的数据被获取，在设计爬虫程序时，通过限制 DFS（深度优先遍历）的深度的方式来间接限定数据与主题的相关性，初步实现过滤部分相关性较低甚至无相关性数据，在数据获取之后，通过对图数据的裁剪，分析，实现数据的二次清洗。

在数据接入方面，通过构建 MySQL 数据库，实现实体属性与实体关系的存储，同时利用其方便的增、删、改、查的功能，方便下一步的数据加工。我们通过调用 MySQLdb 库实现将爬取数据读入 MySQL 数据库。

对象	allItem @kg (emnets) - 表		对象	allItem @kg (emnets) - 表		relationship @kg (emnets) - ...			
开始事务	文本	筛选	排序	开始事务	文本	筛选	排序	导入	导出
item	type		RA	RB	relationship	description			
《有学集》	作品		万历三大征	万历急政	重大战役	明神宗虽不上朝，			
《朱枫林集》	作品		万历三大征	万历朝鲜之役	属于	1592年-1598年万			
一团和气	作品		万历三大征	哮拜	参与	哮拜为蒙古人，素			
一团和气图轴	作品		万历三大征	宁夏之役	属于	宁夏之役归属于万			
一团和气图	作品		万历三大征	播州之役	属于	播州之役归属于万			
一条鞭法	法律		万历三大征	明史	记载	对于三大征的消耗			
丁普郎	历史人物		万历三大征	明朝	重大事件	万历三大征是明神			
丁汝葵	历史人物		万历三大征	明神宗	经历	万历三大征是明神			
丁铎	历史人物		万历三大征	李化龙	参与	播州之役时，明			
万历	历史人物		万历三大征	李如松	参与	万历二十年（159			
万历三大征	战争		万历三大征	杨应龙	参与	播州之役是明朝万			
万历急政	历史事件		万历三大征	邓子龙	参与	万历二十六年（1			
万历朝鲜之役	战争		万历三大征	麻贵	参与	万历十九年（159			
万历野获编	作品		万历朝鲜之役	万历三大征	属于	1592年-1598年万			
万安	历史人物		万历朝鲜之役	东莱城之战	包含	东莱城之战是第一			
万斯同	历史人物		万历朝鲜之役	临津江之战	包含	临津江之战是第一			
三国演义	作品		万历朝鲜之役	临海君	影响	万历二十年（159			
三忠图扇面	作品		万历朝鲜之役	利玛窦	影响	利玛窦联系了北			
三杨	历史人物		万历朝鲜之役	努尔哈赤	影响	然而，万历朝鲜之			
三羊开泰	历史人物		万历朝鲜之役	北关大捷	包含	万历二十年（159			
三藩之乱	历史事件		万历朝鲜之役	北关大捷碑	纪念碑	北关大捷碑，正式			
世界遗产	其它		万历朝鲜之役	南原之战	包含	1597年08月12日			
丘濬	历史人物		万历朝鲜之役	唐浦海战	包含	唐浦海战是万历朝			
丘聚	历史人物		万历朝鲜之役	唐项浦海战	包含	唐项浦海战，又称			
东厂	权力机构		万历朝鲜之役	多大浦之战	开始	多大浦之战，又称			
东厂胡同	地点		万历朝鲜之役	尚州之战	包含	尚州之战是第一次			
东周列国志	历史人物		万历朝鲜之役	平壤之战	包含	1592年06月13日			
东宫三师	地点		万历朝鲜之役	幸州山城之战	包含	幸州山城之战，又			
东林书院	地点		万历朝鲜之役	弹琴台之战	属于	弹琴台之战，又称			
			万历朝鲜之役	明朝	发生于	万历朝鲜之役（朝			

在数据加工方面，从文本数据实现实体识别及关系抽取依靠的是 DeepKE 框架实现。在依据 DeepKE 等框架实现对知识图谱的抽取后，为了实现知识图谱的剪枝以确保知识图谱中实体及关系符合本项目「明代历史知识图谱」的主题，本项目设计了实体及关系的自动剪枝框架，其主要设计思路为利用爬虫在百度百科中爬取每一个实体的

词条信息，并基于语义分析的方式分析该实体与「明代历史」这一主题的关联性，对于与「明代历史」这一主题关联性不强的实体，剪枝程序会剪去此实体及与此实体有关的关系边。

完成数据爬取、知识抽提及知识图谱剪枝后得到的知识图谱中实体-属性信息表及实体间关系信息表分别如上左图及上右图所示。我们利用 MySQL 数据库存储爬虫爬取得到并清洗后的信息，以供之后可视化及 QA 的使用。

2.3 数据可视化

Neo4j 是一个高性能的 NOSQL 图形数据库，它将结构化数据存储在网上而不是表中。它是一个嵌入式的、基于磁盘的、具备完全的事务特性的 Java 持久化引擎，但是它将结构化数据存储在网上（从数学角度叫做图）上而不是表中。Neo4j 也可以被看作是一个高性能的图引擎，该引擎具有成熟数据库的所有特性。程序员工作在一个面向对象的、灵活的网络结构下，而不是严格、静态的表中。但是他们可以享受到具备完全的事务特性、企业级的数据库的所有好处。Neo4j 因其嵌入式、高性能、轻量级等优势，越来越受到关注。

现实中很多数据都是用图来表达的，比如社交网络中人与人的关系、地图数据、或是基因信息等等。RDBMS 并不适合表达这类数据，而且由于海量数据的存在，让其显得捉襟见肘。NoSQL 数据库的兴起，很好地解决了海量数据的存放问题，图数据库也是 NoSQL 的一个分支，相比于 NoSQL 中的其他分支，它很适合用来原生表达图结构的数据。

本项目就基于 Neo4j 实现了项目数据的可视化。我们首先连接 2.2 中数据存储的 mysql 数据库，以便进行数据的读取和后来的建图等步骤，相关代码如下所示。

```
def read_mysql(sql):  
    '''  
        从mysql数据库中数据  
        :param sql: sql查询语句  
        :return: rows 查询结果  
    '''  
    #打开数据库连接  
    dbconn = pymysql.connect(  
        host="xxx.xxx.xxx.xxx", #Your Host IP  
        database='yourDatabase',  
        user='yourUserName',  
        password='yourPassword'  
    )
```

```

#创建游标对象
cur=dbconn.cursor()
# 执行sql语句
cur.execute(sql)
# 获取查询内容
rows = cur.fetchall()
print(rows)
# 关闭数据库连接
cur.close()
dbconn.close()
return rows

```

分别读取已经进行关系抽取与清洗加工的节点数据表和关系数据表里的数据，建立对应的节点和关系边，相关代码片段如下所示。

```

triples = read_mysql("SELECT RA, relationship, RB FROM relationship")
for triple in triples:
    # print network_row
    # 获取节点
    subject_node = nodes[process_string(triple[0])]
    object_node = nodes[process_string(triple[2])]
    relation_type = process_string(triple[1])
    properties = dict()

```

```

# 从MySQL数据库中的types节点导入到neo4j
def get_nodes():
    name_type = dict()
    rows = read_mysql("SELECT distinct item,type FROM allItem")
    for row in rows:
        subject = process_string(row[0])
        subject_type = process_string(row[1])
        name_type[subject] = subject_type
    return name_type

```

最终将节点和关系边写入到 neo4j 数据库中，相关代码如下所示。

```

##### 插入节点到Neo4j中
#####
node_name_type = get_nodes()
nodes = dict()
for (node_name, node_type) in node_name_type.items():
    node = Node(node_type, name=node_name)
    nodes[node_name] = node
    # print ke_node
graph.create(node)

```

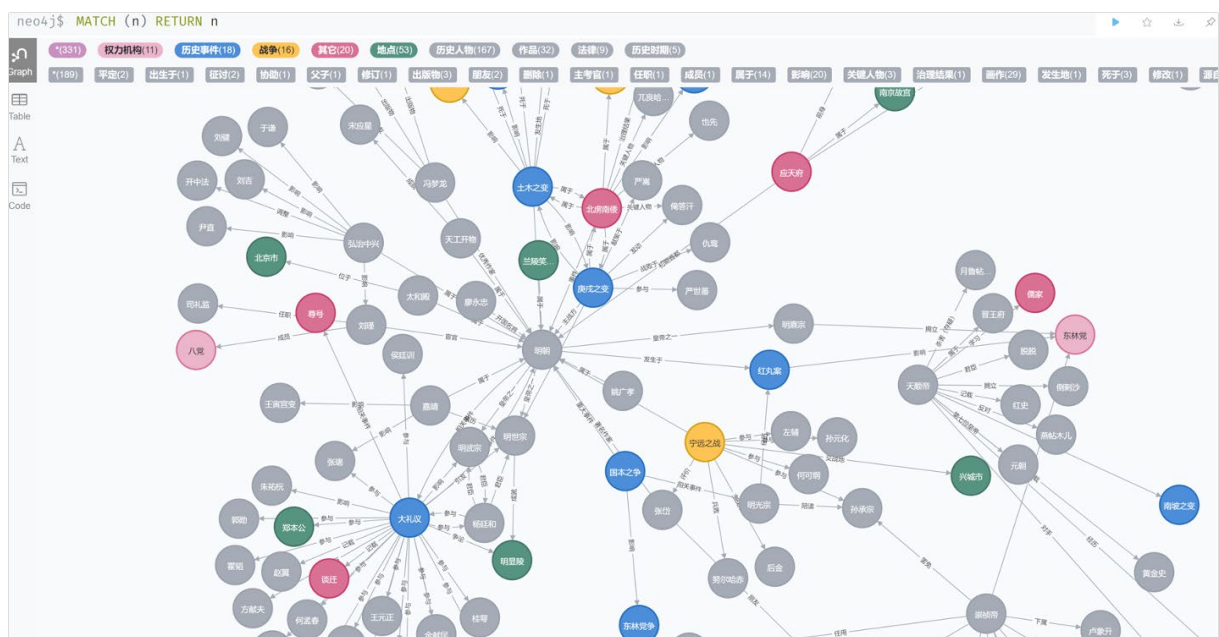
关系创建

```
subject_relation_object = Relationship(subject_node, relation_type,
object_node, **properties)
graph.create(subject_relation_object)
```

由此，本项目的知识图谱在图数据库中的构建已经完成。具体的参数信息如下：
包括 939 个实体节点，1501 条关系边，10 种实体类别和 341 种关系类别。

nodes	939
relationships	1501
labels	10
relationship types	341

将知识图谱可视化如下图所示，因为整个图谱的规模非常庞大，我们只选取了一个子图进行展示。



2.4 图谱应用

我们基于知识图谱中 Neo4j 数据库的 Cypher Query Language 设计了数据查询及问答系统的接口，并基于此实现了明代知识图谱的问答。Cypher 是一个描述性的图形查询语言，允许不必编写图形结构的遍历代码对图形存储有表现力和效率的查询。对于使用者的问题，一个可资参考的 Cypher 语言解析案例如下所示。

问句	解析结果	Cypher 语句
朱棣的父亲是谁	question_type:"relationship_query",args:{ nodes:["朱棣"],relationship:["父子"]}	MATCH(n:relationship{name:"朱棣"})->[r:父子] return r.父子

解析过程中技术要点如下：其一，匹配关键词：由于领域内有较多专有名词，传统分词工具分词效果较差，因此本项目中使用了匹配外部字典的方式检测关键词，并存储至相应参数；其二，解析疑问词：采用建立字典的形式，将常见的疑问词与对应的语义建立连接以利于 Cypher 语言查询；其三，根据疑问词及关键词匹配问题类型，传递相应参数，构造 Cypher 语言实现查询。

三、项目文件组织

本项目文件共分为若干部分，包括由用于构建知识图谱的原始数据及基于 Neo4j 构建的知识图谱文件，其组织形式如下所示。

```
- history-knowledge-graph-neo4j.dump    #从Neo4j导出的项目dump文件
- KG.py                                #用于连接SQL数据库并操纵Neo4j图数据库生成知识图谱
- rawData                              #用于构建知识图谱的原始数据，包含实体详情、实体间关系及关系属性
  - allItem.json    #所有用于构建知识图谱的实体信息
  - allRelationship.json #知识图谱中所有实体间关系信息
  - relationship.json    #记录知识图谱中各个实体间关系及关系属性
- MingDynastyHistory.sql #用于生成记录知识图谱各实体及其关系的SQL数据库
```

四、总结

在本次的项目实战开始前，小组成员充分讨论，顺应历史要求与时代潮流，一致同意选定当前明代历史人物关系知识图谱这一主题。在实战过程中各成员分工明确，各司其职，共同完成了包括信息抽取、知识融合加工等步骤在内的整个知识图谱的构建过程，不仅加深了小组成员对知识图谱的理解、提升了自己的专业水平，还对明代历史人物的关系有了不同程度的了解。我们一致认为，经过了本次的项目实战，我们在小组合作、专业水平、文化素养等方面都收获颇多！