《哈利·波特》人物关系问答系统说明文档

项目成员：
组长：安嘉骏 20110127
组员：杨佳悦 20110108 李精文 20110115 黄丁彬 20110123 麦热姆 20110160

1 引言

1.1 项目名称

基于知识图谱的《哈利·波特》人物关系问答系统

1.2 项目背景

《哈利·波特》是英国作家 J.K.罗琳于 1997～2007 年所著的魔幻文学系列小说，共 7 部。据哈利·波特官方网站不完全统计，截至 2002 年，前 4 部在英国和美国共销售了 1.75 多亿册。截至 2013 年 5 月，“哈利·波特”系列丛书已经被翻译成 73 种语言，共卖出了超过 5 亿本。在中国，人民文学出版社于 2000 年 8 月推出了前三部的中译本，首印 60 万册，社长聂展宁在采访中透露仅一年半的时间中，《哈利·波特》图书就销售了 7700 万码洋，创下建国以来单本图书出版的奇迹。2007 年 10 月 28 日中文版第七部上市第一天便销售了 12500 册，又一次刷新了单本图书单日销售码洋纪录。由此可见，“哈利·波特”系列丛书在全世界范围内影响广大。

此外，“哈利·波特”系列丛书具有结构庞大、出现人物繁多和人物关系复杂等特点。这类特点使得故事更具有可信度和完整性，然而也在一定程度上增加了读者阅读的困难。读者无法轻易记住书中出现的每一个人物姓名或是人物关系，这可能导致读者对一段情节描述产生困惑。因此，设计一种《哈利·波特》人物关系问答系统，以实现帮助读者回忆或理解书中人物关系的功能，对提升读者阅读体验有重要意义。

1.3 任务目标

本项目将构建小说《哈利·波特》中的各种实体及其关系的知识图谱与问答系统。首先基于一哈利·波特相关内容 wiki 网页，爬取其已有的各种实体关系，通过数据清洗后，构建初步的知识图谱三元组。防止网页文本信息不全，我们又使用预训练的实体识别模型从小说原文中抽取现有实体关系，对先前构造的三元组进行扩充，构建最终的知识图谱。接下来将内容存入 Neo4j 图数据库中，使用 ltp 语言分词、词性标注功能构建问答系统后端。最终使用 Flask 搭建简易 python 网页服务器，使用 Bootstrap 框架编写网页前端，最终搭建起哈利·波特人物关系问答系统。

1.4 参考资料

[1]李小龙,孙水发,唐庭龙,耿骞,吴义熔.基于超声检查报告的乳腺癌诊断知识图谱构建[J/OL].武汉大学学报(理学版):1-10[2022-12-08].

[2]钱涛,卢方超,韩梓政,戴文华.基于知识图谱的豆瓣读书问答系统[J].湖北科技学院学报,2022,42(06):147-151+156.

2 需求分析

2.1 功能概述

本系统基于知识图谱相关知识，通过搭建好的哈利·波特人物关系问答系统，用户能够使用自然语言向问答系统提问人物关系，问答系统能够对自然语言进行处理，提取用户问题，自动生成数据库查询语言，对数据库进行查询，最终返回给用户匹配结果，用户无需在繁杂的网页或小说中寻找其关系。

2.2 运行环境

硬件环境：Windows/MacOS/Linux
开发软件：Pycharm 2022.3
Neo4j 5.2.0

2.3 假定与约束

（1）用问答系统进行提问时需输入人物名的全称，如哈利·波特。

（2）由于人物之间的关系种类繁多，本项目对关系进行分类汇总，共包含以下关系，查询时若查询失败，尝试查找已有关系进行查询。

图 1 全部关系种类

（3）使用本程序需要在本地部署好相关环境，并保持互联网连接。

3 逻辑设计

基于知识图谱的问答系统主框架如图 2 所示，主要包括三个模块：知识图谱构建模块、问题分析模块和答案查询模块。

在知识图谱构建模块，主要分为知识抽取、知识推理、知识融合以及知识可视化这 4 步。其中，我们分别采用了半结构化数据（网站）和纯文本数据（原著）进行知识抽取，以保证知识的全面性。将两组关系数据进行融合，并将融合后的三元组导入图数据库中。

在问题分析模块，用户通过前端页面输入问题，由后端对问题进行语义分析并分词，然后进行词性标注，最终提取出问句核心，生成查询语句，在图数据库上查询答案并返回结果。

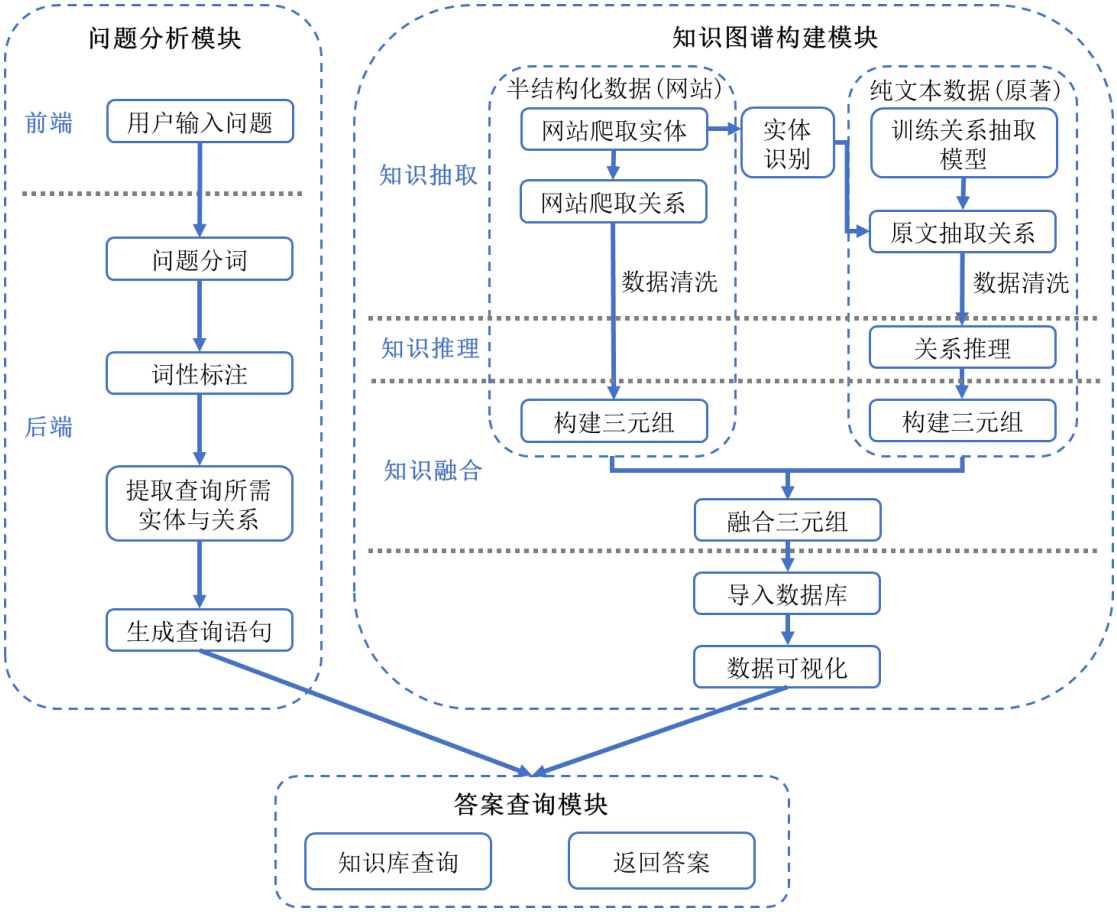

图 2 基于知识图谱的问答系统框架图

4 详细设计

4.1 半结构化数据处理

数据获取与预处理是项目的第一部分，旨在获取后期需要处理的所有数据。在本项目中，这一步是通过爬取中文维基网站获取哈利波特中的实体名称，进而通过实体名称进行爬取实体关系数据。最后对实体以及实体关系进行初步清理。

4.1.1 实体获取

获取实体的主要是通过实体所在类别进行检索所有实体数据，爬取的链接：https://harrypotter.fandom.com/zh/wiki/Category：*。这里的*为实体类别。步骤是先手动定义部分实体类别，通过检索第一个类别中含有的实体链接，筛选出链接类别是成员链接的，成员链接可能为实体链接、模板链接、实体所在的类别链接。进而将实体所在类别链接对应的类补充到手动创建的类别，把实体链接对应的实体记录。

如图 3，以麻瓜类为例，链接为 https://harrypotter.fandom.com/zh/wiki/Category:麻瓜，爬取该网页的所有实体链接。

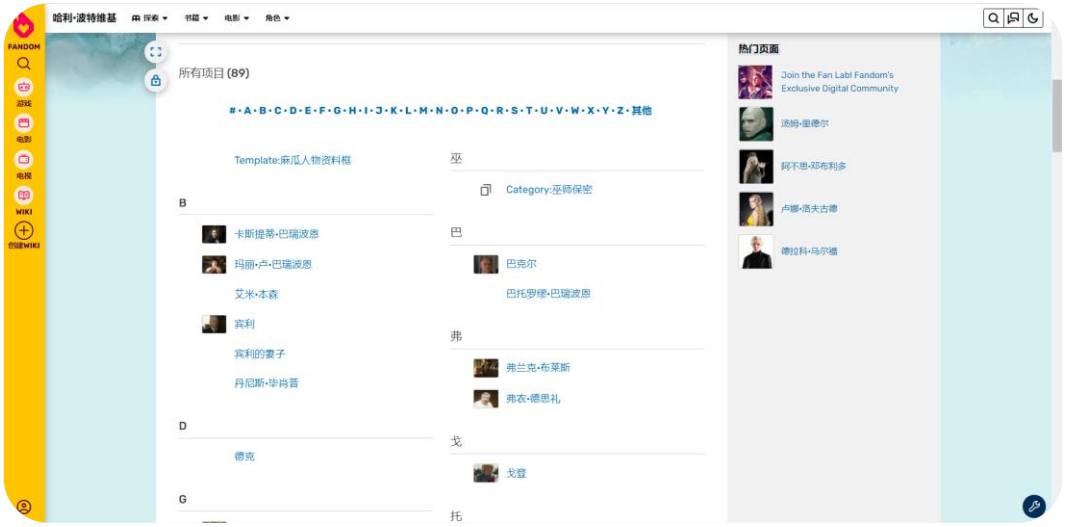

图 3 《哈利·波特》麻瓜类网页

对应的代码为 extract_raw_entities.py 文件

4.1.2 实体关系获取

通过上一步提取出来的实体，爬取对应网站里面的关系列表，从而获得三元组数据。

图 4 哈利·波特人物关系列表

如图 4，以爬取哈利·波特这一人物的关系为例，通过右侧家庭信息表格进行关系爬取。爬取后的关系三元组如图 5 所示。

乔治·韦斯莱	从属	韦斯莱魔法把戏坊
乔治·韦斯莱	儿子	弗雷德·韦斯莱二世
乔治·韦斯莱	双胞胎哥哥	弗雷德·韦斯莱
乔治·韦斯莱	叔叔	比利尔斯
乔治·韦斯莱	哥哥	查理·韦斯莱
乔治·韦斯莱	哥哥	比尔·韦斯莱
乔治·韦斯莱	哥哥	珀西·韦斯莱
乔治·韦斯莱	外祖母	普威特夫人
乔治·韦斯莱	外祖父	普威特先生

图 5 爬取得到的关系三元组

对应的代码为 extract_raw_relations.py 文件

4.1.3 数据清理

数据清理方法主要是删除不规则实体、删除不规则实体关系和拆分并列的实体关系。关系三元组中有错误的信息，如图 6 所示，需要进行删除。

卢修斯·马尔福	岳父	西格纳斯·布莱克三世
卢修斯·马尔福	父亲	阿布拉克萨斯·马尔福
卢修斯·马尔福	祖先	卢修斯·马尔福一世
卢修斯·马尔福	祖先	布鲁图斯·马尔福
卢修斯·马尔福	祖先	阿曼德·马尔福
卢修斯·马尔福	职业	(?-1996年)
卢修斯·马尔福	职业	(? - 1993年)
佩妮·德思礼	职业	(曾经)
卢修斯·马尔福	职业	霍格沃茨魔法学校董事会
卢修斯·马尔福	职业	魔法部官员
卢多·巴格曼	从属	(曾经)
卢多·巴格曼	从属	巴格曼家庭
卢多·巴格曼	从属	温布恩黄蜂队
卢多·巴格曼	从属	魔法部
卢多·巴格曼	弟弟	奥托·巴格曼
卢多·巴格曼	父亲	巴格曼先生
卢多·巴格曼	职业	(曾经)
卢多·巴格曼	职业	温布恩黄蜂队击球手

图 6 关系三元组中的错误信息（部分）

对应的代码为 clean_relations.py 文件

4.2 非结构化数据处理

4.2.1 实体识别

（1）文件结构：

对应文件位于项目根目录 relation_extraction 文件夹下，文件名：extract_relation_and_clean.py

（2）实现过程：

①原文分句

将哈利波特原文 Harry_Potter.txt 每一行头尾的空格或换行符移除，将非空的每一行存储到列表 total_lines 中。

```
with open("Harry_Potter.txt", "r", encoding="utf-8") as f:
    total_lines = [line.strip() for line in f.readlines()]
#列表推导式，移除txt文件每一行头尾的空格或换行符。将处理后的每一行存储到列表total_lines中。
total_lines = [line for line in total_lines if line != '']
# 将非空的每一行存储到列表total_lines中。
```

图 7 获取原文中的每行

将"？ ", "！ ", "。 ", "!"作为分句的标志，对列表 total_lines 中每行进行遍历进行分句。

```
# 分句
cutLineFlag = ["？", "！ ", "。", "!"]
#将"？", "！", "。", "!"作为分句的标志
sentenceList = []
#处理后的句子列表
for words in total_lines:#对处理后的每行进行遍历
    oneSentence = ""#赋值
    for word in words:#对处理后的每行中的词进行遍历
        if word not in cutLineFlag:#判断词中是否含cutlineFlag的五种符号
            oneSentence = oneSentence + word#将oneSentence赋值为新增word后的不完全句子
        else:
            oneSentence = oneSentence + word
            if oneSentence.__len__() > 4:#如果oneSentence中的元素大于4
                sentenceList.append(oneSentence.strip())#去除头尾的空格或换行符，添加到处理后的句子列表
            oneSentence = ""#重新赋值
```

图 8 每行进行分句

②获取所有的实体

从半结构化数据中获取所有实体名称，读取 harryid.csv 文件到 DataFrame，放入元组 csv 中，取出所有实体放入列表 origin_id。

```
csv = pd.read_csv("data/harryid.csv", header=0)#读取CSV文件到DataFrame，设定header=0替换掉原来存在列名，放入元组csv中
origin_id = list(csv.iloc[:, 1])#取所有行索引，列索引为1的数据放入未处理的实体列表中。
```

图 9 获取所有实体名称

通过遍历获取实体全名和本人名（最后的名），放入列表 total_id。

```
total_id = []#处理后的实体列表
for id in origin_id:#对未处理的实体列表数据进行遍历
    total_id.append(id)#将每个全名添加到处理后的实体列表
    for sub_id in id.split('.'):#用split对每个全名取本人名（最后的名）
        total_id.append(sub_id)#将本人名添加到处理后的实体列表
```

去除列表 total_id 中的重复与空格

```
total_id = list(set(total_id))#通过set函数处理为为一个无序不重复的列表
total_id.remove('')#移除空格
```

图 10 去除重复与空格

③匹配原文中的实体

对处理句子列表 total_lines 进行遍历，判断每个实体列表 total_id 中的实体是否在句子中出现，若出现则对其与出现的句子进行匹配，返回查询到实体在句子中所在位置的起始和结束位置，并放入列表 loc 中，将能匹配到的实体放入列表 id_list，匹配到的位置放入 id_loc。

```
for sentence in sentenceList:#对处理好的句子列表进行遍历
    id_loc = []#匹配到的实体位置列表
    id_list = []#匹配到的实体列表
    for id in total_id:#对处理好的实体列表进行遍历
        if id in sentence:#实体是否在该句子中
            loc = [(item.start(), item.end()-1) for item in re.finditer(id, sentence)]
            #将实体与句子进行匹配，返回查询到实体在句子中所在位置的起始和结束位置，并放入列表loc中
            id_list.append(id)#将能匹配到的实体放入该表中
            id_loc.append(loc[0])#将匹配到的实体在句子中的位置放入该表中
            # print(id_loc)
```

图 11 匹配原文中的实体

当同一个句子能够有多个实体匹配到时，对这些实体两两组合，将不重复的实体与该句子作为新的关系数据项，一起放入列表 new_date 中。

```
if len(id_loc) >= 2:#当匹配到的位置数据大于等于二时
    permute = list(itertools.combinations(range(len(id_list)), 2))
    #创建一个长度为能匹配到的实体数量的整数列表，并实现两个整数间的两两组合，并放入表permute中
    # print(len(permute))
    for idx in permute:
        if id_list[idx[0]] not in id_list[idx[1]] and id_list[idx[1]] not in id_list[idx[0]]:
            new_data.append({'text':sentence, 'h': {'pos': id_loc[idx[0]]}, 't': {'pos': id_loc[idx[1]]}})
        #如果两个实体不重复，则新增这两个实体关系数据项，放入列表new_data中
```

图 12 匹配到多个实体

4.2.2 关系抽取

我们基于 OpenNRE 框架训练关系抽取模型，然后将《哈利·波特》原文进行分句，并根据此前在哈利波特维基上爬取的实体名筛选出含有多个实体的句子，使用关系抽取模型抽取每个句子中实体间的关系，最后对得到的关系进行数据清洗，并将新增的人物关系数据扩充到通过网站爬取得到的三元组中。

①预训练数据集

（1）文件结构：

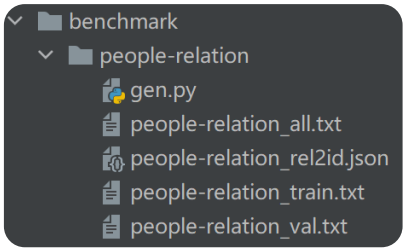

图 13 关系抽取预训练数据集文件结构

其中，gen.py 可用于生成人物关系数据，people-relation_rel2id.json 存储了 12 种关系，people-relation_train.txt 和 people-relation_val.txt 分别为训练集和验证集。

（2）数据描述：

我们使用 https://github.com/taorui-plus/OpenNRE 提供的人物关系数据集。该数据集中的人物关系包括：父母、夫妻、师生、兄弟姐妹、合作、情侣、祖孙、好友、亲戚、同门、上下级和 unknown，共 11+1 种关系。选取其中一个语料如下：

{"token": ["第", "一", "个", "上", "场", "的", "曹", "云", "金", "和", "刘", "云", "天", "合", "说", "《", "口", "吐", "莲", "花", "》", "，", "　", "接", "着", "郭", "德", "纲", "和", "于", "谦", "表", "演", "了", "相", "声", "《", "我", "要", "上", "春", "晚", "》", "、"], "h": {"name": "刘云天", "pos": [10, 12]}, "t": {"name": "曹云金", "pos": [6, 8]}, "relation": "合作"}

图 14 关系抽取预训练数据举例

其中，h 和 t 后面分别代表两个实体，pos 是实体在句子中的位置，relation 是两者之间的关系。

②模型训练

（1）文件结构：

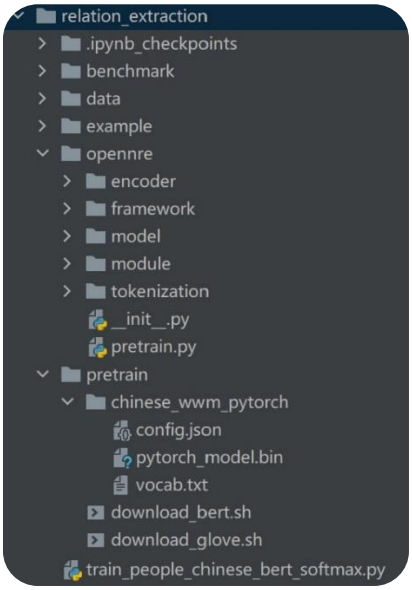

图 15 模型训练文件结构

其中，train_people_chinese_bert_softmax.py 为训练模型文件，部分关键代码如下图所示。

```
# Define the model
model = opennre.model.SoftmaxNN(sentence_encoder, len(rel2id), rel2id)

# Define the whole training framework
framework = opennre.framework.SentenceRE(  # 普通句子级别的关系抽取流程 sentence_re
    train_path=os.path.join(root_path, 'benchmark/people-relation/people-relation_train.txt'),
    val_path=os.path.join(root_path, 'benchmark/people-relation/people-relation_val.txt'),
    test_path=os.path.join(root_path, 'benchmark/people-relation/people-relation_val.txt'),
    model=model,
    ckpt=ckpt,
    batch_size=64,  # Modify the batch size w.r.t. your device
    max_epoch=3,
    lr=2e-5,  # learning rate
    opt='adamw'  # Adam + weight decate
)

# Train the model
framework.train_model()

# Test the model
# load_state_dict()函数用于将预训练的参数权重加载到新的模型之中
framework.load_state_dict(torch.load(ckpt)['state_dict'])
result = framework.eval_model(framework.test_loader)
```

图 16 模型训练文件部分代码

（2）训练过程：

在准备阶段，首先需要在 https://github.com/ymcui/Chinese-BERT-wwm 下载 chinese_wwm_pytorch 模型（不包含 MLM 任务的权重），并放置在./pretrain/下。其中，Whole Word Masking (wwm)是谷歌在 2019 年发布的一项 BERT 的升级版本，优化了原预训练阶段的训练样本生成策略。放置完模型后，还需要配置环境变量：openNRE=项目位置，或在./opennre/pretrain.py 中将 default_root_path = os.path.join(os.getenv('openNRE'), '.')改成自己的目录路径。

其次，由于环境和部分依赖包版本不同，需要对 OpenNRE 框架的源代码做部分修改，在./opennre/encoder/bert-encoder.py 中：

```
_, x = self.bert(token, attention_mask=att_mask)
```

改为：

```
_, x = self.bert(token, attention_mask=att_mask, return_dict=False)
```

在这里需要设置 bert 的返回值不是字典类型，否则会出现如下报错：

```
  File "D:\pythonwork\harrypotter\relation_extraction\opennre\model\softmax_nn.py", line 46, in forward
    rep = self.drop(rep)
  File "D:\Python\Python38\lib\site-packages\torch\nn\modules\module.py", line 1190, in _call_impl
    return forward_call(*input, **kwargs)
  File "D:\Python\Python38\lib\site-packages\torch\nn\modules\dropout.py", line 59, in forward
    return F.dropout(input, self.p, self.training, self.inplace)
  File "D:\Python\Python38\lib\site-packages\torch\nn\functional.py", line 1252, in dropout
    return _VF.dropout_(input, p, training) if inplace else _VF.dropout(input, p, training)
TypeError: dropout(): argument 'input' (position 1) must be Tensor, not str
```

图 17 报错信息

准备阶段完成后，即可运行 train_people_chinese_bert_softmax.py 训练 BERT 中文关系抽取模型。

③模型测试

（1）对应文件：

extract_relation_and_clean.py

```
from tqdm import tqdm
relation_list = []
for data in tqdm(new_data):
    text = data['text']
    t_pos = data['t']['pos']
    h_pos = data['h']['pos']
    rela = model.infer(data)  # 调用模型抽取关系
    # 调用模型的infer函数，传递(1)一个段落，(2)第一个实体位置，以及(3)第二个实体位置。该函数返回实体对的预测关系，使用段落作为上下文。
    relation_list.append([text[t_pos[0]:t_pos[1]+1], text[h_pos[0]:h_pos[1]+1], rela])
```

图 18 调用模型抽取关系代码

（2）具体操作：

在该阶段首先对《哈利波特》原文进行分句，并根据此前在哈利波特维基上爬取的实体名筛选出含有多个实体的句子，标注出实体及实体位置。然后导入上一阶段训练好的关系抽取模型，抽取每个句子中的关系。

然而，我们发现通过模型识别出的关系仍有错误，因此需要进一步对关系进行清洗。

④关系数据清洗与关系推理

（1）对应文件：

extract_relation_and_clean.py

（2）具体操作：

相比于关系抽取模型中的关系，网站上爬取的关系数据中的亲属关系更为详细，因此我们这里主要关注师生、合作等关系。基于此，我们做了如下操作：

1.由于不同句子可能会包含相同的实体对，导致最终可能一个实体对会存在多个关系，我们选择概率最高的一组作为最终的关系；

2.句子中的实体对还可能存在歧义，我们使用候选实体中在文章中出现频率最多的实体来替换；

3.手动剔除一些明显的错误，得到最终的关系数据；

4.对已抽取得到的关系进行简单推理，例如：

A 师生 B 关系能倒退回 B 学生 A 关系

A 同门 B B 同门 C 能推理出 A 同门 C

4.3 知识融合

（1）对应文件：

relation_fusion.py

```
def remove_repeated(data):  # 去重
    res = set()  # 集合去重
    for d in data:
        res.add((d[0],d[1],d[2]))
    return sorted(list(res))
```

图 19 去除重复的关系数据代码

（2）具体操作：

将经过实体识别与关系抽取得到的三元组与最初网站上爬取得到的三元组进行融合，增加额外的三元组，合并相同的三元组，得到最终的数据（relations_final.csv）如图 20 所示。

```
C.沃林顿,斯莱特林魁地奇球队,从属
C.沃林顿,调查行动组,从属
C.沃林顿,霍格沃茨魔法学校,从属
乔治·韦斯莱,亚瑟·韦斯莱,父亲
乔治·韦斯莱,凤凰社,从属
乔治·韦斯莱,吉迪翁·普威特,舅舅
乔治·韦斯莱,哈利·波特,妹夫
乔治·韦斯莱,塞德瑞拉·布莱克,祖母
乔治·韦斯莱,塞德里克·迪戈里,同门
乔治·韦斯莱,塞普蒂默斯·韦斯莱,祖父
```

图 20 最终得到的三元组

4.4 可视化与知识问答

4.4.1 图数据库构建

（1）依赖：

Neo4j Desktop Version 1.5.6：构建哈利波特人物关系图数据库

py2neo 2021.2.3：辅助实体关系导入

（2）文件结构：

构建文件位于项目根目录 neo_db 文件夹下，具体包含如下文件：

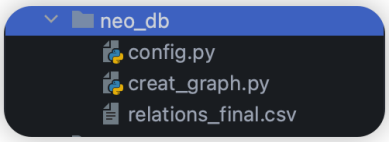

图 21 图数据库文件结构

其中，relations_final.csv 为将要导入的任务关系文件，内容文件格式为：[实体，实体，关系]

以下为部分内容：

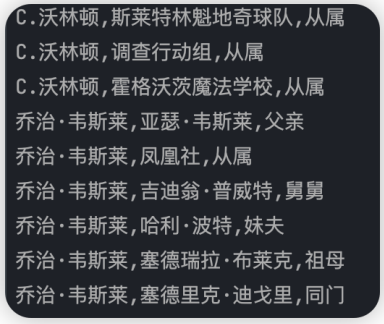

图 22 任务关系文件

config.py 文件为 Neo4j 图数据库连接配置文件，包含数据库连接地址、用户名、密码等信息，使用时由用户自行配置。

```
from py2neo import Graph
graph = Graph(
    "http://localhost:7474",  # 数据库连接地址
    name="neo4j",  # 数据库用户名
    password="123456"  # 数据库密码
)
```

图 23 图数据库连接配置文件

creat_graph.py 文件作用为将 relations_final.csv 数据批量导入至数据库，具体内容如下：

```python
with open("relations_final.csv") as f:  # 打开将要导入数据库的csv文件
    for line in f.readlines():          # 逐行读取
        rela_array = line.strip("\n").split(",")  # 将回车去除后，以逗号为分界分隔文本为array
        print(rela_array)  # 将文本打印在终端中显示
        rela_array = [rela_array[0], rela_array[-1], rela_array[1]]  # 变更array顺序
        graph.run("MERGE(p: Entity{Name: '%s'})" % (rela_array[0]))  # 若没有该实体，创建
        graph.run("MERGE(p: Entity{Name: '%s'})" % (rela_array[2]))  # 若没有该实体，创建
        graph.run(
            "MATCH(e: Entity), (cc: Entity) \
            WHERE e.Name='%s' AND cc.Name='%s'\
            CREATE(e)-[r:%s{relation: '%s'}]->(cc)\
            RETURN r" % (rela_array[0], rela_array[2], rela_array[1], rela_array[1])
        )  # 将两个实体的节点建立关系
```

图 24 数据库导入文件

（3）构建过程：

首先在主机端安装好 Neo4j Desktop 图数据库，并设定好用户名与密码。

将 config.py 中的数据库地址、用户名、密码依照个人的设置配置好。

使用如下命令运行 create_graph.py 文件，将实体关系导入图数据库：

> python3 create_graph.py

该部分运行完成后，即可创建好图数据库

在数据库页面使用如下指令查询所有导入数据

> MATCH (n) RETURN (n)

结果如下:

图 25 数据导入可视化

使用如下指令，查询哈利·波特的老师

> MATCH p=(n:Entity{Name:"哈利·波特"})-[r:`学生`]->() RETURN p;

结果如下：

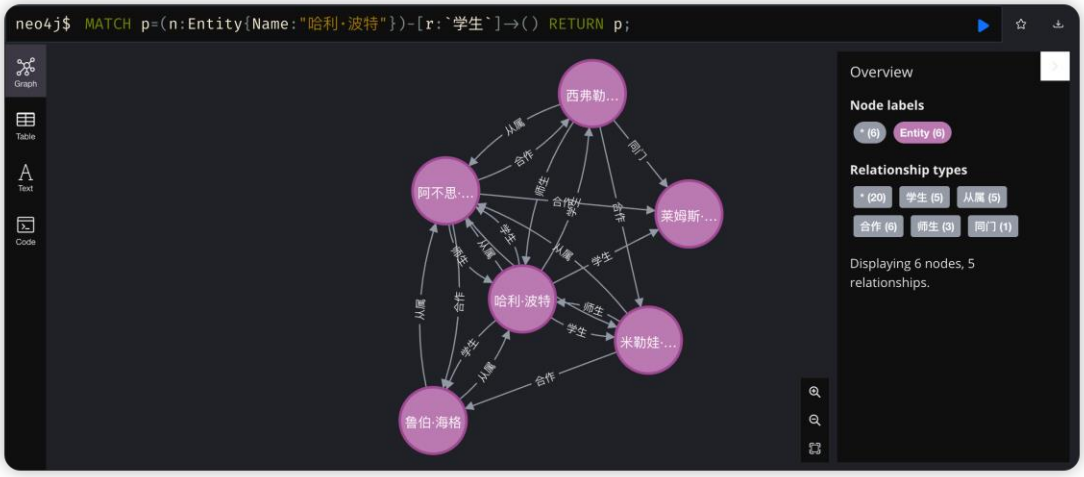

图 26 数据导入可视化

至此，图数据库建立完成。

4.4.2 问答系统后端搭建

（1）依赖：

pyltp 0.4.0 :用于处理分词模型与词性标注

ltp_data_v3.4.0：分词模型，需要下载到本地部署，下载地址在代码注释中已提供

（2）文件结构：

该部分文件位于根目录 kgqa 包下，包含 4 项文件，其中 ltp_data_v3.4.0 为分词模型。

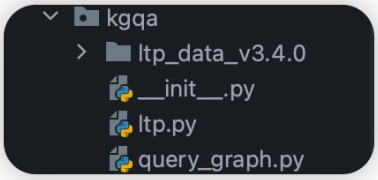

图 27 问答系统后端文件结构

ltp.py 用于问答系统用户输入问题进行语义分析并提取。

```python
def cut_words(words):
    # 分词模型，模型名称为`cws.model`
    segmentor = pyltp.Segmentor(os.path.join(LTP_DATA_DIR, 'cws.model'))  # 初始化实例
    words = segmentor.segment(words)  # 分词
    array_str = "|".join(words)  # 使用竖线分隔
    array = array_str.split("|")  # 分隔后变为array
    segmentor.release()  # 释放模型
    return array
```

图 28 语义分析提取文件

分词模型将句子分为词列表，如：乔治·韦斯莱的叔叔是谁？分词后为['乔治·韦斯莱', '的', '叔叔', '是', '谁', '？']

再使用词性标注模型

```
def words_mark(array):
    # 词性标注模型路径，模型名称为`pos.model`
    postagger = pyltp.Postagger(os.path.join(LTP_DATA_DIR, 'pos.model'))  # 初始化实例
    postags = postagger.postag(array)  # 词性标注
    pos_str = ' '.join(postags)  # 使用空格分隔
    pos_array = pos_str.split(" ")  # 分隔后变为array
    postagger.release()  # 释放模型
    return pos_array
```

图 29 词性标注模型

对应分词结果进行词性标注，结果为

['nh', 'u', 'n', 'v', 'r', 'wp']

最终提取出问句核心，构建列表为

['乔治·韦斯莱', '叔叔', '谁']

query_graph.py 用于将上述问句核心转换为 cypher 语句，进行查询图数据库操作。

该部分代码量较多，在文件代码中有详细介绍其作用的注释，此处不再赘述。

（3）使用过程：

在项目根目录下的 qa_app.py 可直接运行后端程序。

通过打开 cmd 或 terminal 等工具，切换到项目根目录下，运行以下代码，即可进入命令行主程序进行查询。

> python3 qa_app.py

```
~/Downloads/harrypotterkgcodes  python3 qa_app.py
/Users/anjiajun/Downloads/harrypotterkgcodes/kgqa/ltp_data_v3.4.0/cws.model
欢迎使用哈利·波特人物关系问答系统
This is an example:
Q:乔治·韦斯莱的叔叔是谁？
A:
比利尔斯

请输入您的问题，输入quit可退出
==========================================================================================
Q:
```

图 30 问答系统后端查询

此时输入问题内容，如：乔治·韦斯莱的父亲是谁？得到如下结果：

```
欢迎使用哈利·波特人物关系问答系统
This is an example:
Q:乔治·韦斯莱的叔叔是谁？
A:
比利尔斯

请输入您的问题，输入quit可退出
==========================================================================================
Q:乔治·韦斯莱的父亲是谁？
A:
亚瑟·韦斯莱
==========================================================================================
Q:
```

图 31 问答系统后端查询

输入 quit 即可退出后端程序。

4.4.3 问答系统前端搭建

（1）依赖：

Flask2.2.2：用于使用 python 构建简易网页服务器

（2）文件结构：

根目录下的 main.py 为 flask 构建文件。

templates 为网页 html 存放位置，其中包括一个 index.html 为前端页面。

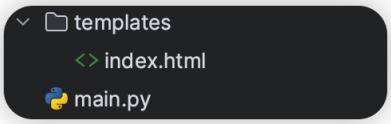

图 32 问答系统前端文件结构

（3）使用过程：

使用 cmd 或 terminal 等命令行操作工具，切换到项目根目录下，执行

> export FLASK_APP=main

接着执行

> flask run

此时，若无报错，则通过浏览器进入 http://127.0.0.1:5000，即可看到问答系统页面。

图 33 问答系统前端界面

若报错 5000 端口被占用，则运行以下命令，更改至空端口即可

> flask run --port=5001

在查询区域输入问题：

图 34 问答系统前端查询

点击 search 后返回结果

图 35 问答系统前端查询结果

5 所用方法与技术

5.1OpenNRE

OpenNRE 是清华大学推出的开源关系抽取框架，针对命名实体识别，句子级别的关系抽取，远程监督（包级别的关系抽取），文档级别的关系抽取以及 few-shot 任务均有实现。OpenNRE 在实现时，将关系抽取的框架划分成不同的模块，其模块化的设计可以大幅度减少不必要的代码重写，其整体架构如图 36 所示。

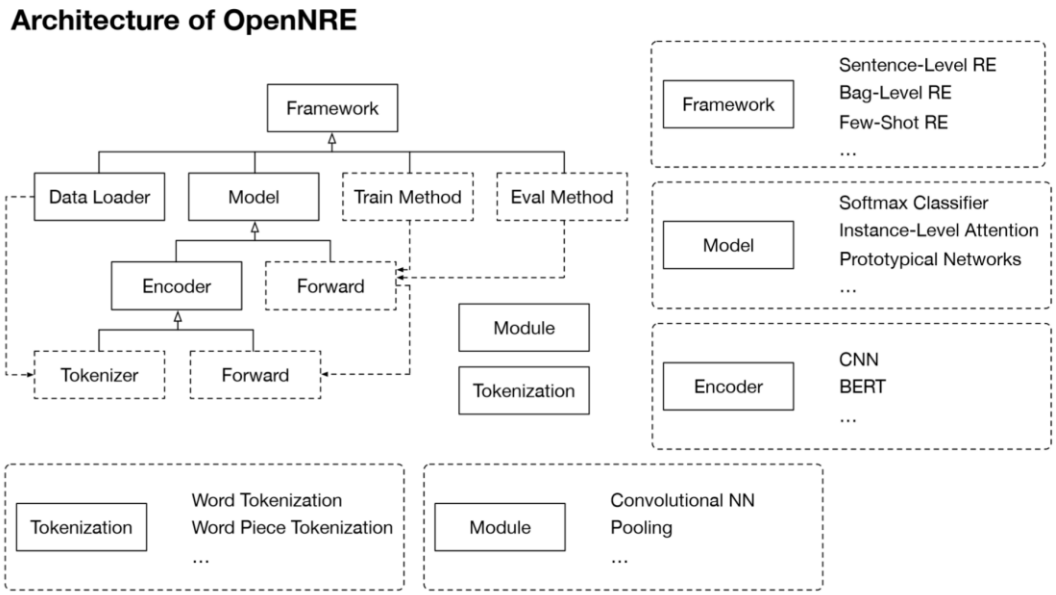

图 36 OpenNRE 整体架构

在 DataLoader 模块中，针对不同的任务，实现了不同的读取策略和 DataSet 类。

在 Encoder 模块中，实现了各个模型获得关系向量表示的步骤。如用 PCNN+最大池化得到关系向量表示，BERT 模型的特殊标记[CLS]作为关系向量表示，将两个实体前面插入特殊标记E_{1start},E_{2start}，并将这两个特殊标记的隐藏向量拼接作为关系向量表示（BERTem 模型）等。

在 Model 模块中，实现了在获取关系向量表示后的分类策略。如普通的全连接加 softmax，远程监督的将一个包中所有的关系向量平均作为包的关系向量表示再过全连接和 softmax，以及对于包中的实例应用 attention 策略后得到向量表示再进行分类等。

在 Train Method 和 Eval Method 中是一些稍加改动就可以直接使用的训练步骤。

FrameWork 模块是对上述所有模块进行集成，包括普通句子级别的关系抽取流程 sentence_re ，以及远程监督包级别的关系抽取 bag_re 等。

5.2LTP

LTP 提供了一系列中文自然语言处理工具，用户可以使用这些工具对于中文文本进行分词、词性标注、句法分析等等工作。从应用角度来看，LTP 为用户提供了下列组件：

（1）针对单一自然语言处理任务，生成统计机器学习模型的工具

（2）针对单一自然语言处理任务，调用模型进行分析的编程接口

（3）系统可调用的，用于中文语言处理的模型文件

（4）针对单一自然语言处理任务，基于云端的编程接口

该工具可实现中文分局、用户自定义词典、分词、词性标注、命名实体识别、语义角色标注、依存句法分析、语义依存分析。

在本应用中，使用 LTP 构建问答系统的语义识别功能，主要使用了 LTP 的分

词与词性标注功能。

6 场景实现

图 37 问答系统场景实现

具体内容详见实际操作视频。