《斗破苍穹》小说知识图谱

黄志伟 22321285　陈信宇 22321181　刘益斌 22321358　刘镕琛 12321247

一、项目介绍

本项目构建了《斗破苍穹》小说的知识图谱，数据主要来源于《斗破苍穹》百度百科与《斗破苍穹》小说原文。构建本图谱的工作主要分为以下几个部分：

1. 数据获取与预处理
2. 关系抽取
3. 属性提取
4. 构建问答系统

二、数据获取与预处理

2.1 数据爬取

在数据爬取部分，我们从百度百科和 wiki 爬取有关斗破苍穹小说的人物、剧情等信息，作为训练集语料库。我们使用 requests_html 库对网页进行访问，并爬取网页的 html 文件，再通过 BeautifulSoup 库对网页进行解析，提取出有用的信息。爬取过程具体如下：

首先需要从百度百科的“斗破苍穹”小说词条中获取小说中出现的人物名、异火名和势力名等信息，并提取这些词条的 url，以便后续的访问。获得以上 url 之后，就需要对这些 url 进行单独访问，提取每一个词条的信息，百度百科的词条一般由简介部分和正文部分组成，需要将爬取下来的内容进行处理，使其成为结构化的数据。

爬取数据如图所示：

2.2 构建实体列表

由于网络小说中的实体名称与现实存在较大区别，而百度百科中已经有较为全面的实体介绍，且整个图谱涉及的实体类型较少，因此我们采用基于匹配的实体识别策略。首先需要从爬取数据中提取实体名称与类型，构建实体列表。

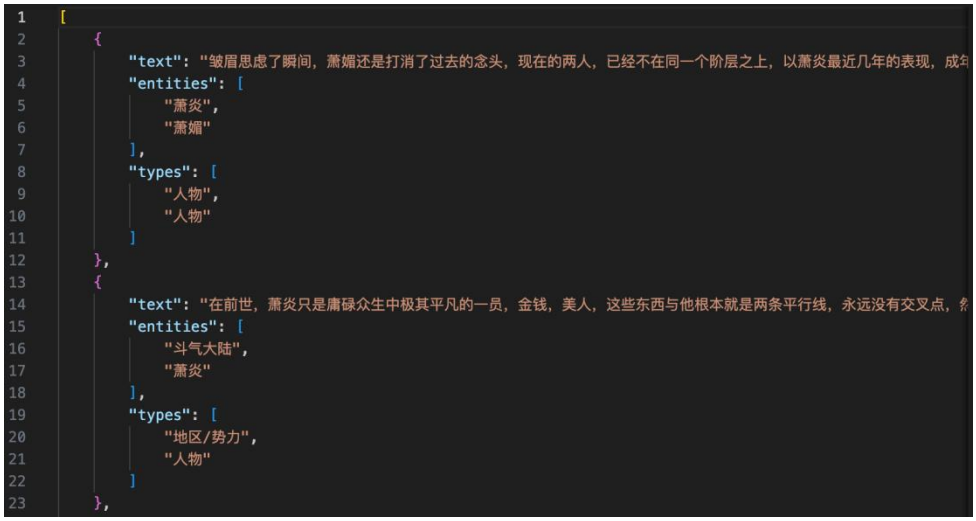
需要注意的是在《斗破苍穹》中，同一个人物可能会有不同的称呼，比如“药尘”可能会被叫做“药老”、“药尊者“，在构建实体列表时需要将每个人物的称呼统一，在后面对原文文本预处理中也需要对称呼进行替换统一。

2.3 数据预处理

根据构建的实体列表对原文文本进行预处理，具体步骤如下所示：

1. 去除原文文本中作者相关内容；
2. 对原文文本中的人名称呼进行统一；
3. 对文本按照标点符号进行分句；
4. 从实体列表中进行匹配，保留存在两个及以上实体的句子；
5. 去除一些过短句子，构建原文数据集。

构建的原文数据集如图所示：



三、关系抽取

3.1 基于 LLM 的关系抽取

近期火热的 LLM 在文本理解与对话问答上表现出了极大的潜力，能够从给定文本中按照要求执行指定的任务。因此，对于文本内容较为复杂的网络小说，我们也尝试使用 LLM 来进行关系的抽取。DeepKE 中已经提供了基于 LLM 的关系抽取工具，其背后封装了一套 prompt 模板，能够将我们的上下文、实体、候选关系组合成一组问答指令，发送给 ChatGPT，指导其从上下文中提取实体关系。

3.1.1 prompt 模板修改

prompt 模板封装在 EasyInstruct 工具中，在使用中我们发现了当前 prompt 存在的一些问题，并对 prompt 模板进行了一些修改。

首先，在需要提取关系的上下文中可能并不能体现实体之间的关系，因此如果无法提取关系则指示 GPT 输出“unknown”。此外，在提问时，对 GPT 做一些加强的指示能够提高 GPT 的回答质量。因此我们修改 prompt 模板如下：

```
# RE_CH_LABELS = "您是一个高度智能和精确的关系抽取（RE）系统。给定上下文以及上下文中包含的一对头实体和尾实体，您的任务是提取给定头实体和尾实体间特定类型的关系，候选的关系类型如下：\n{0}.\n"
RE_CH_LABELS = "您是一个高度智能和精确的关系抽取（RE）系统。给定上下文以及上下文中包含的一对头实体和尾实体，您的任务是提取给定头实体和尾实体间特定类型的关系。请仔细理解所给上下文，注意关系是单向的，不清楚则回复unknown。候选的关系类型如下：\n{0}.\n"
```

修改完成后，需要重新安装 EasyInstruct 工具，执行命令行代码：pip install -e .

3.1.2 数据生成

由于 GPT 使用成本限制，我们并不使用小说原文进行关系抽取，而是使用从百度百科中爬取的描述语句进行。百度百科中的人物实体都有“介绍”内容，我们将上下文组织成“(实体) 是 (人物介绍)”的形式，作为上下文，如下所示：

```
# 人物：萧潇
# 介绍：萧炎与彩鳞的女儿，实力十分强悍，出生即斗宗强者，在萧炎晋升为斗帝时，成为八星斗圣。在《大主宰》中与牧尘相遇。
# 上下文：萧潇是萧炎与彩鳞的女儿，实力十分强悍，出生即斗宗强者，在萧炎晋升为斗帝时，成为八星斗圣。在《大主宰》中与牧尘相遇。
```

我们已经有了实体的列表，使用匹配的方法从上下文中识别其他实体信息，并分别与主实体构成头尾实体进行关系抽取，比如上图中的“萧潇”与“萧炎”将进行一次关系抽取。上下文中的所有实体均将进行一次抽取。

3.1.3 关系抽取

数据生成后，调用 easyinstruct 接口构建 prompt：

```
prompt = ie_prompter.build_prompt(
                prompt=role['text'],
                head_entity=entity1,
                head_type=entity1_type,
                tail_entity=entity2,
                tail_type=entity2_type,
                language='ch',
                instruction=None,
                in_context=False,
                domain=None,
                labels=relations,
                examples=None
            )
```

其中 prompt 输入的是用于抽取的上下文内容，此外还需要指定头尾实体及其类型、候选的关系列表 relations。候选关系列表如下所示：

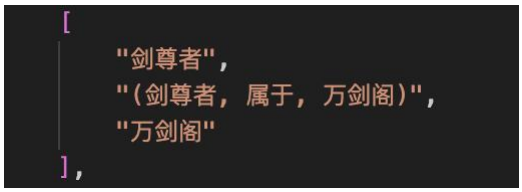
```
relations = ['父亲', '母亲', '儿子', '女儿', '哥哥', '弟弟', '妹妹', '姐姐', '伙伴', '妻子', '丈夫', '喜欢', '师父', '徒弟', '敌对', '属于', '统治', '拥有', '位于', '占领', '控制', '仇人', 'unknown']
```

然后我们调用 get_openai_result 接口以获取 GPT 回复，并以三元组的形式存储结果。考虑到成本与时间问题，我们选择使用 gpt-3.5-turbo 模型进行关系抽取。

3.1.4 三元组清洗

使用 LLM 提取的三元组中，同样会存在一些错误三元组，需要进行清洗与处理。尽管已经将 GPT 的 temperature 设置为 0，但是在有些问答中会出现没有按照要求格式回复的情

况需要人工处理，如下所示：



此外，一些错误的关系或者需要合并的关系可以通过人工进行处理，最后构成基于 LLM 提取的关系图谱。

3.2 基于 DeepKE 的关系抽取

DeepKE 是一个开源的知识图谱抽取工具，可以通过定制输入的数据集和模型实现命名实体识别、关系抽取和属性抽取功能。在本项工作中主要应用该工具对相关文本进行实体识别与关系抽取。由于原文文本过多，考虑到基于 LLM 提取的成本过高，我们尝试自行微调预训练模型从原文中提取额外的关系。

3.2.1 数据集构建

对于命名实体识别工作，我们使用此前获得的实体列表构建词汇表，对小说文本与使用从百度百科中爬取的描述语句进行查表标注，从而获取较大规模的训练数据集；对于关系抽取工作，我们在确定模型训练使用的关系列表后，使用 GPT 提取出的关系与人工标注的关系构建训练集。为便于模型训练，我们将部分关系进行合并，并将句子中未标注关系的实体的关系标注为“无关”。最终使用的关系列表如下：

```
[
    {
        "人物":{
            "人物":["敌人","伙伴","夫妻","父母","子女","师父","徒弟","喜欢","祖辈","孙辈","兄姊","弟妹","红颜"],
            "地区/势力":["属于","敌对势力","统治","创建"],
            "异火":["拥有"]
        },
        "地区/势力":{
            "人物":["门人/族人","敌对者","首领","创建者"],
            "地区/势力":["位于","附属","同盟","敌对"],
            "异火":["掌握"]
        },
        "异火":{
            "人物":["被拥有"],
            "地区/势力":["掌握于"],
            "异火":[]
        }
    }
]
```

由于我们标注得到的数据集标签数量较为不平衡，我们结合人工添加语句、过采样与欠采样让数据集相对平衡以便于后续训练。

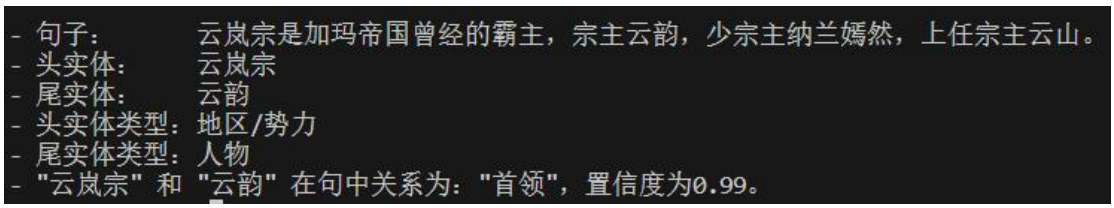
3.2.2 实体识别

我们使用 DeepKE 的命名实体识别模型，用准备好的数据集在基于 BERT 的模型上进行训练，测试结果如下：

在训练好实体识别模型后，可以用该模型提取给定句子含有的实体名称与类别，可以充当基于匹配的实体识别的补充识别方案，用于识别其他不在匹配列表中的实体。

3.2.3 关系抽取

我们使用 DeepKE 的关系抽取模型，用 GPT 与人工标注筛选好的数据集在基于 BERT+LSTM 的模型上进行训练，测试结果如下：



我们将句子与从实体识别模型中识别所得的实体内容一起输入训练好的 re 模型，对一个句子中的每一个实体对进行关系抽取。在获取模型提取的关系数据后，我们对这些数据进行人工数据清洗，并添加到知识图谱中。

四、属性提取

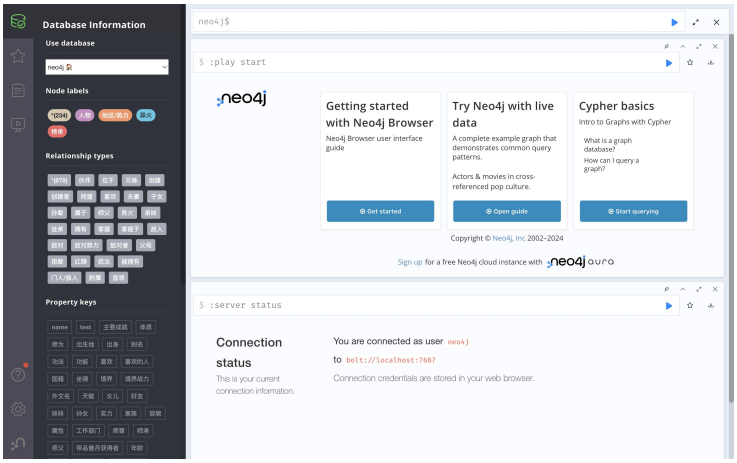
在属性提取部分，需要提取人物和异火的相关属性（例如人物的别名、性别和异火的颜色、排行等），在属性提取部分，可以直接使用百度百科简介中的结构化数据来完成。在数据的处理部分，需要筛选掉一些异常的属性，以及与小说无关的属性，如由小说改编的电视剧的相关信息等。最终提取的属性类似于字典的格式，被写入 json 文件保存。属性的内容和结构如下所示：


```
"萧炎": {
    "中文名": "萧炎",
    "别名": "药岩、岩枭、炎盟之主、大主宰",
    "性别": "男",
    "年龄": "数百岁",
    "称号": "炎帝",
    "灵力境界": "主宰境",
    "灵魂境界": "帝境（斗破时期）"
},
```

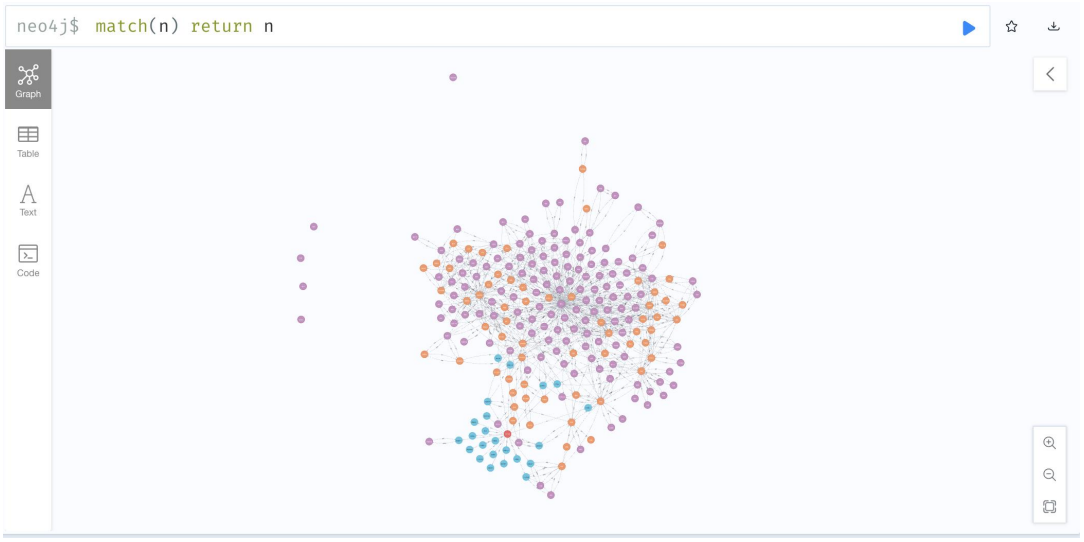

五、可视化与知识问答

5.1 知识图谱可视化

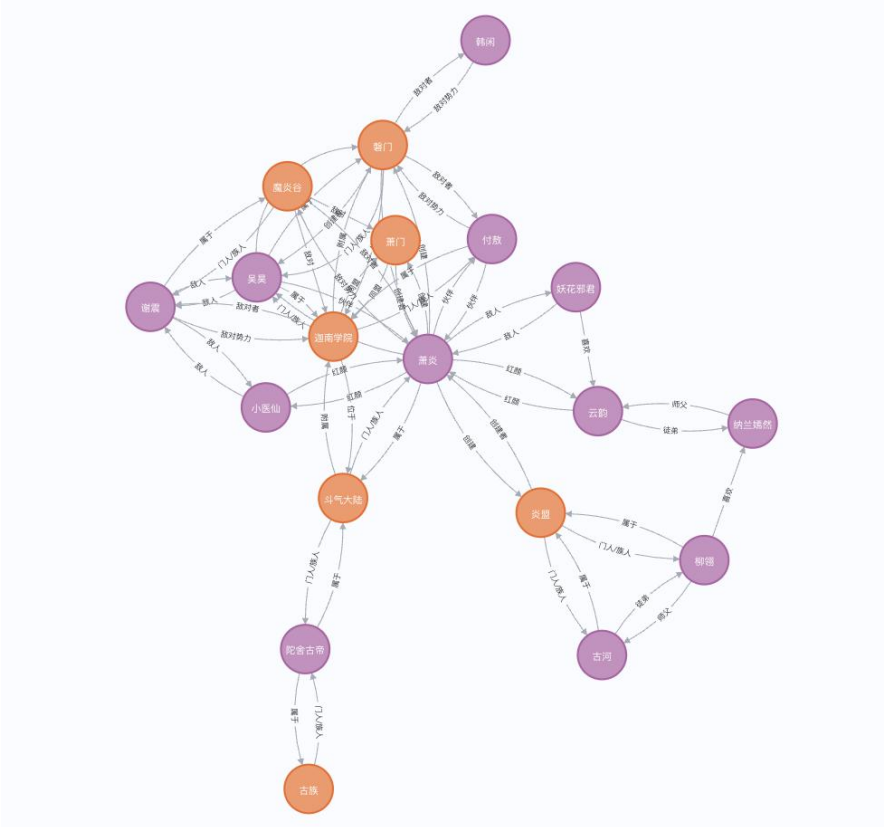
通过 Neo4j Desktop，我们对《斗破苍穹》中对提取出来对实体、关系以及属性可视化，我们将前面导出对 dump 文件导入到 neo4j 中并且创建一个数据库用于存放该数据。导入完数据之后即可启动该项目并且进入到对应对数据库中进行操作：



我们通过 match(n) return n 查看整个知识图谱：



图中的节点和属性较多，因此我们通过知识查询可视化与主角萧炎相关的 25 条关系：



5.2 知识问答

基于 Cyther 查询，我们利用 jieba 的 paddle 模式对输入对查询句子进行分词，将第一个提取对名次词语作为实体，第二个名次词语作为关系，利用 answer()函数实现问题解析。该问答系统支持人物关系对查询：

```
def answer(self, sentence):
    """
    :param sentence:str 用户输入的问句，如"萧炎的哥哥是谁"
    :return result: str 如: ""
    """
    try:
        sentence = re.sub("[A-Za-z0-9\!\%\[\]\,\。]", "", sentence)
        sentence = re.sub('\W+', '', sentence).replace("_", '')
        person, words = self.cut_words(sentence)
        words_ = [0 for i in range(len(words))]
        words_new = [0 for i in range(len(words))]
        for i in range(len(words)):
            words_new[len(words)-i-1] = self.neo.similar_words.get(words[len(words)-i-1],
words[len(words)-i-1])
            if isinstance(words_new[len(words)-i-1], list):
                words_[i] = '-[r'+str(i) + ':'+words_new[len(words)-i-1][0]+']'
            else:
                words_[i] = '-[r'+str(i) + ':'+words_new[len(words)-i-1]+']'
            if i != len(words)-1:
                words_[i] += '->(n'+str(i)+':Person)'
        if isinstance(words_new[0], list):
            quary = "match(p:人物{name:'"+person +\
                "'})" + ''.join(words_) + \
            "->(n) where n.性别 = '"+words_new[0][1]+"' return  p.name,n.name"
        else:
            quary = "match(p:人物{name:'"+person + \
                    "'})" + ''.join(words_) + \
                "->(n) return  p.name,n.name"
        logging.debug(str(quary))
    except Exception as e:
        return '问题有误'
    try:
        data = self.neo.graph.run(quary)
        data = list(data)
        logging.debug(str(data))
        result = data[0]['p.name']+'的'
        for item in words:
            result += item
        result +="是"
        for i in range(len(data)):
            result += data[i]['n.name']
            if i!=len(data)-1:
                result +=","
        return result
    except Exception as e:
        return '没有答案'

```

比如，输入问题：萧炎的哥哥是谁？后，首先会将句子分词并且提取到名次萧炎和哥哥，将萧炎作为实体，哥哥作为关系，由于我们的关系中并没有定义哥哥，而是定义的兄姊，因

此实际上是将查询的关系设置为兄姊，并且需要设置条件 `性别`=`男`。接着再用构成的查询语句去数据库中查找，得到的结果为：

```
(KG) liuyibin@liuyibindeMacBook-Air QA % python qa.py
Paddle enabled successfully......
[2024-01-13 21:29:05,827] [   DEBUG] _compat.py:47 - Paddle enabled successfully......
问：萧炎的哥哥是谁
答：_萧炎的哥哥是萧厉,萧鼎
```

其他的查询结果展示：

```
问：萧炎的伙伴有哪些
答：  萧炎的伙伴是凌影,吴昊,林修崖,琥嘉,柳擎,林焱,叶欣蓝,米特尔·腾山,邝天尺,曜天火,古灵,唐火儿,黑擎,妖暝,付敖,铁乌,苏媚,阴骨老人,金石,萧玉,萧薰儿
问：萧炎的师父是谁
答：  萧炎的师父是若琳,药尘
问：萧炎的老婆有哪些
答：  萧炎的老婆是萧薰儿,彩鳞
问：萧炎的敌人有哪些
答：  萧炎的敌人是魂殿殿主,北龙王,费天,妖蛇夏莽,洪辰,沈云,鸷鹰,铁护法,蜈崖,慕兰三老,云帆,雷纳,罗侯,沙铁,苏笑,范崂,蒙喇,罗布,穆力,穆蛇,九凤,魂玉,药星极,妖花邪君,黑、白天尊,摘星老鬼,慕骨,青海,雷尊者,凤清儿,墨承,翎泉,韩枫,云山
问：雷尊者的徒弟有谁
答：  雷尊者的徒弟是凤清儿
问：萧薰儿的父亲是谁
答：  萧薰儿的父亲是古元
问：萧玉的弟弟有谁
答：  没有答案
问：萧玉的哥哥有谁
答：  萧玉的哥哥是萧宁
```