

浙江大学



基于《盗墓笔记》小说的知识图谱构建与应用

知识图谱导论课程设计

题 目：	基于《盗墓笔记》小说的知识图谱构建与应用
上课时间：	周五3,4
授课教师：	陈华均
姓 名：	赖述忠，方美静，陈雨萌，王振阳
日 期：	2020.07.04

基于《盗墓笔记》小说的知识图谱构建与应用

赖述忠，方美静，陈雨萌，王振阳

浙江大学计算机学院

简介：本报告介绍了基于《盗墓笔记》的知识图谱的构建过程和智能问答系统的设计。通过从维基百科和相关知乎回答获取数据，利用DeepKE进行实体识别和关系抽取，我们成功构建了包含人物、地点、物品等实体以及它们之间的关系的知识图谱。此外，我们还开发了基于知识图谱的智能问答系统，能够根据用户提出的问题从知识图谱中获取准确的答案。

关键词：知识图谱；盗墓笔记；智能问答系统；openKE

1 项目简介

背景与动机：《盗墓笔记》是一部极具影响力的小说系列，其丰富的故事情节和独特的世界观吸引了大量读者和粉丝。为了更好地理解小说中的人物、事件和情节，构建一个《盗墓笔记》的知识图谱是十分必要和有意义的。通过知识图谱技术，可以将小说中的关键信息以结构化的方式呈现出来，为读者提供更深入、更全面的阅读体验，同时也为相关研究和探索提供了便利。

项目目标：本项目旨在利用知识图谱技术，基于《盗墓笔记》小说的内容，构建一个包含实体、关系和属性的知识图谱，并通过可视化和问答系统等方式，使用户能够更直观地探索和理解《盗墓笔记》的世界。

项目内容与方法：

- 数据获取与预处理：**从《盗墓笔记》维基百科和相关知乎回答等来源获取原始数据，并进行清洗、去重等预处理工作，以便后续的知识图谱构建。
- 实体识别与关系抽取：**使用deepKE等工具进行实体识别和关系抽取，从原始文本中提取出实体和实体之间的关系，构建知识图谱的基础。
- 属性提取：**进一步从文本中提取实体的属性信息，如人物的特点、事件的背景等，丰富知识图谱的内容。
- 知识图谱可视化：**利用可视化工具将构建的知识图谱呈现出来，以图形化的方式展示《盗墓笔记》中的人物、事件和关系，提升用户的交互体验。
- 构建问答系统：**基于构建的知识图谱，设计和实现一个问答系统，使用户能够通过提问获取关于《盗墓笔记》的相关信息，提高信息获取的效率和便利性。

2 数据获取与预处理

本部分代码是《盗墓笔记》维基百科页面的爬取与数据提取过程。通过该代码，我们能够从维基百科页面中提取出关于人物介绍、老九门以及相关名词的内容，并将其保存为CSV文件，以便后续的数据处理和分析。

爬取维基百科页面：使用Python的requests库发送HTTP请求，获取《盗墓笔记》维基百科页面的HTML内容。

```
url = "https://zh.wikipedia.org/wiki/%E7%9B%9C%E5%A2%93%E7%AD%86%E8%A8%98"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
```

提取与转换内容： 利用BeautifulSoup库解析HTML内容，提取出页面中包含的文本内容，并进行繁简体转换。

```
content = soup.find("div", {"id": "mw-content-text"})
content_text = content.get_text()
simplified_content = chinese_converter.to_simplified(content_text)
```

提取人物介绍、老九门和名词内容： 通过正则表达式匹配，从转换后的文本中提取出人物介绍、老九门和名词部分的内容。

```
pattern1 = r"人物介绍\[编辑\]\n([\s\S]*?)\n老九门\[编辑\]"
match_results1 = re.search(pattern1, simplified_content)
if match_results1:
    results1 = match_results1.group(1)

pattern2 = r"老九门\[编辑\]\n([\s\S]*?)\n名词\[编辑\]"
match_results2 = re.search(pattern2, simplified_content)
if match_results2:
    results2 = match_results2.group(1)

pattern3 = r"名词\[编辑\]\n([\s\S]*?)\n出版书籍\[编辑\]"
match_results3 = re.search(pattern3, simplified_content)
if match_results3:
    results3 = match_results3.group(1)
```

保存为CSV文件： 将提取出的内容保存为CSV格式的文件，以便后续的处理和分析。

```
with open("wiki_人物.csv", "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    for row in results1_rows:
        writer.writerow([row])

with open("wiki_老九门人物.csv", "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    for row in results2_rows:
        writer.writerow([row])
```

```

with open("wiki_名词.csv", "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    for row in results3_rows:
        writer.writerow([row])

```

以上是数据获取部分的介绍，通过这些代码，我们能够快速、有效地从《盗墓笔记》维基百科页面中提取出关键信息，并保存为结构化的数据文件，为后续的知识图谱构建奠定了基础。对于知乎页面的处理类似，不再加以赘述。

3 实体识别、关系提取和属性提取

主要是完成两个工作：

- 实体识别：讲文本中的特定实体进行识别
- 关系预测：预测分句中任意两个实体间的关系

首先将从Wiki百科中爬取的数据进行汇总和格式的统一，将所有的实体和关系，分别加载到entity.txt，relation.txt中

```

import pandas as pd
import jiagu
import jieba
import requests
import re
import time

data_people = pd.read_csv("wiki_人物清单.csv")
data_nine = pd.read_csv("wiki_老九门人物清单.csv")
data_object = pd.read_csv("wiki_名词清单.csv")

def write2txt(data):
    with open('entity.txt', 'a+', encoding='utf-8') as f:
        for index, row in data.iterrows():
            f.write(row[0])
            f.write('\n')

```

然后尝试利用jiagu库来直接抽取三元关系

```

def split_sentence(text):
    # 按句号拆分文本成分句
    sentences = text.split('。')
    # 打印分句结果
    return sentences

def extration_relation_jiagu(df):

```

```

jiagu.load_userdict('entity.txt')
for index, row in df.iterrows():
    sentences = split_sentence(row[0])
    print(sentences)
    for sentence in sentences:
        words = jiagu.seg(sentence) # 分词
        print(words)

        pos = jiagu.pos(words) # 词性标注
        print(pos)

        ner = jiagu.ner(words) # 命名实体识别
        print(ner)

        knowledge = jiagu.knowledge(sentence)
        print(knowledge)

```

由于《盗墓笔记》是书籍，所以其实体是特殊的，书中特有的，所有直接抽取的效果并不好

所以我们利用 [zjunlp/DeepKE](#)这个基于深度学习的开源中文知识图谱抽取框架来完成实体识别和三元组抽取的过程，我们利用DeepKE-cnSchema借助预训练好的模型 **DeepKE(NER), RoBERTa-wwm-ext, Chinese** 来完成实体识别任务，**DeepKE(RE), RoBERTa-wwm-ext, Chinese** 来完成三元组抽取的任务。

DeepKE提供了预训练好的支持cnSchema的特别版DeepKE-cnSchema，支持开箱即用的中文实体抽取和关系抽取等任务，可抽取50种关系类型和28种实体类型，其中实体类型包含了通用的人物、地点、城市、机构等类型，关系类型包括了常见的祖籍、出生地、国籍、朝代等类型。

将DeepKE-main/example/triple/cnschema/conf中的predict.yaml中的模型位置进行适当的修改

```

nerfp: 'DeepKE-main/example/triple/cnschema/data/nerfp'
refp: 'DeepKE-main/example/triple/cnschema/data/re_bert.pth'

```

然后运行DeepKE-main/example/triple/cnschema/predict.py

```
python predict.py
```

就可以利用上述的两个预训练模型来自动的进行实体识别和三元组抽取

同时为了测试抽取的效果，我们将爬取的人物、物品的描述按句号分句，然后将分句分别送入模型中，

最终得到的实体命名结果和三元组抽取效果会比没有分句前要好

但是上述的总体抽取效果并不好，首先语料并不丰富，而且模型并没有在《盗墓笔记》的专用训练集上进行预训练和微调，所以抽取的效果并不能达到构建知识图谱并进行后续问答的水平，所以我们利用 [APIs \(fudan.edu.cn\)](#) 中提供的API，然后根据我们预先收集好的实体清单，来获取与其相关的三元组信息，并进行相应的清洗和整合

获取实体相关信息

```
def getEntity(entity):
    url = 'http://shuyantech.com/api/cndbpedia/ment2ent?q='

    url = url + entity

    # GET请求示例
    response = requests.get(url)

    # 处理API响应数据
    if response.status_code == 200:
        # API请求成功
        # 进一步处理响应数据
        response_data = response.json()
        print(response_data)
    else:
        # API请求失败
        print('API请求失败:', response.status_code)
    return response_data
```

获取三元组相关信息

```
def getRelation(entity):
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0',}

    url = 'http://shuyantech.com/api/cndbpedia/avpair?q='

    url = url + entity

    response_flag = False
    # api容易timeout, 需要多次重试
    while True:
        # time.sleep(1)
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            print('Get response')
            response_flag = True
            data = response.json()
            print(data)
        else:
            print('No response, Try again')
```

```

        if response_flag == True:
            break

    return response.json()

```

将得到的三元组以[head, type, tail]的方式存储

```

def get_entity_from_api(file_path):
    entity_dataset = []
    entity = get_entity_from_txt(file_path)
    for i in entity:
        reponse_entity = getEntity(i)
        for j in reponse_entity['ret']:
            entity_dataset.append(j)
    return entity_dataset

def get_relation_from_api(file_path):
    relation_dataset = []
    entity = get_entity_from_txt(file_path)
    for i in entity:
        print(i)
        reponse_relation = getRelation(i)
        for j in reponse_relation['ret']:
            j.insert(0, i)
            print(j)
            relation_dataset.append(j)
    return relation_dataset

```

由于三元组关系中存在DESC属性（对于实体的大段描述），所以我们将该属性的内容添加到原来Wiki百科中爬取的内容部分，并从知乎上爬取了故事的梗概，送入DeepKE中进行相应的抽取

我们发现得到的三元组关系中有一部分是人物关系，另外一部分是人物属性，所以我们对其进行了区分，方便后续的可视化建模和问答系统的构建

同时对type进行了归类，将所有包含‘名’的类型都替换为‘别名’，并删除了一些不必要的关系和实体，

最终将这里筛选得到的三元组与原来DeepKE上得到的三元组合并，得到最终用于构建知识图谱的
relation_triple.xlsx和attr_triple.xlsx

遍历每一行并修改指定列的值

```
for index, row in attr_triple.iterrows():
    if '名' in [j for j in row['type']]:
        attr_triple.at[index, 'type'] = '别名'
attr_triple = pd.DataFrame(columns=['head', 'type', 'tail'])
for index, row in relation_df.iterrows():
    type_of_triple = row['type']
    if type_of_triple in attr_txt:
        attr_triple.loc[len(attr_triple)] = row
```

处理得到的文件描述如下：

最终的三元组文件：

- attr_triple_clear.xlsx : 属性的三元组抽取（进行人工筛选）
- relation_triple.xlsx: 关系的三元组抽取
- attr_triple.xlsx : 属性的三元组抽取

附加文件：

- attribution.txt: 所有的属性一览
- entity.txt: 所有的实体一览
- type.txt: 所有的类型（属性+关系）一览
- relation.txt: 所有的关系一览
- raw_data.xlsx: Wiki百科+DESC描述在DeepKE上的三元关系抽取结果
- zhihu.txt: 知乎总结的txt
- zhihu.xlsx: 知乎总结上的三元关系抽取结果

然后根据可视化需要的json，将xlsx文件转换为json文件

```
attr = pd.read_excel('attr_triple_clear.xlsx')
attr_json = {}
for en in entity:
    filter_data = attr[attr['head']==en]
    temp_json = {}
    for index, row in filter_data.iterrows():
        temp_json[row['type']] = [row['tail']]
    attr_json[en] = temp_json
attr_json = json.dumps(attr_json, ensure_ascii=False)
with open('attr.json', 'w', encoding='utf-8') as json_file:
    json_file.write(attr_json)
```

处理得到的文件描述如下：

- relation.json : 所有的关系

- nodes.json : 所有的实体分类
- links_and_nodes.json : relation + nodes
- attr.json : 实体的属性

links_and_nodes.json

```
{
  "links": [
    {
      "relation": "母亲",
      "source": "张起灵",
      "target": "白玛（藏医）",
      "value": 3
    },
    ...
  ],
  "nodes": [
    ...
  ],
  ...
}
```

attr.json

```
{
  "吴邪": {
    "职业": [
      "开古董店"
    ],
    "生日": [
      "1977年3月5日(双鱼座)"
    ],
    "身高": [
      181
    ],
    "体重": [
      "65KG"
    ],
    "武器": [
      "黑金匕首"
    ],
    "性别": [
      "男"
    ]
  },
  ...
}
```

4 知识图谱可视化

在可视化过程中，我们主要采用实体关系数据进行构建，但由于选取的数据量不大，因此在最后关系划分时可能会遇到一些问题，比如关系中的目标实体在实体集中找不到对应的情况。为了解决这个问题，我们进行了以下处理：

```
python
import json

# 读取原始的links_and_nodes.json文件
with open('links_and_nodes.json', 'r', encoding='utf-8') as f:
    content = f.read()
    data = json.loads(content)

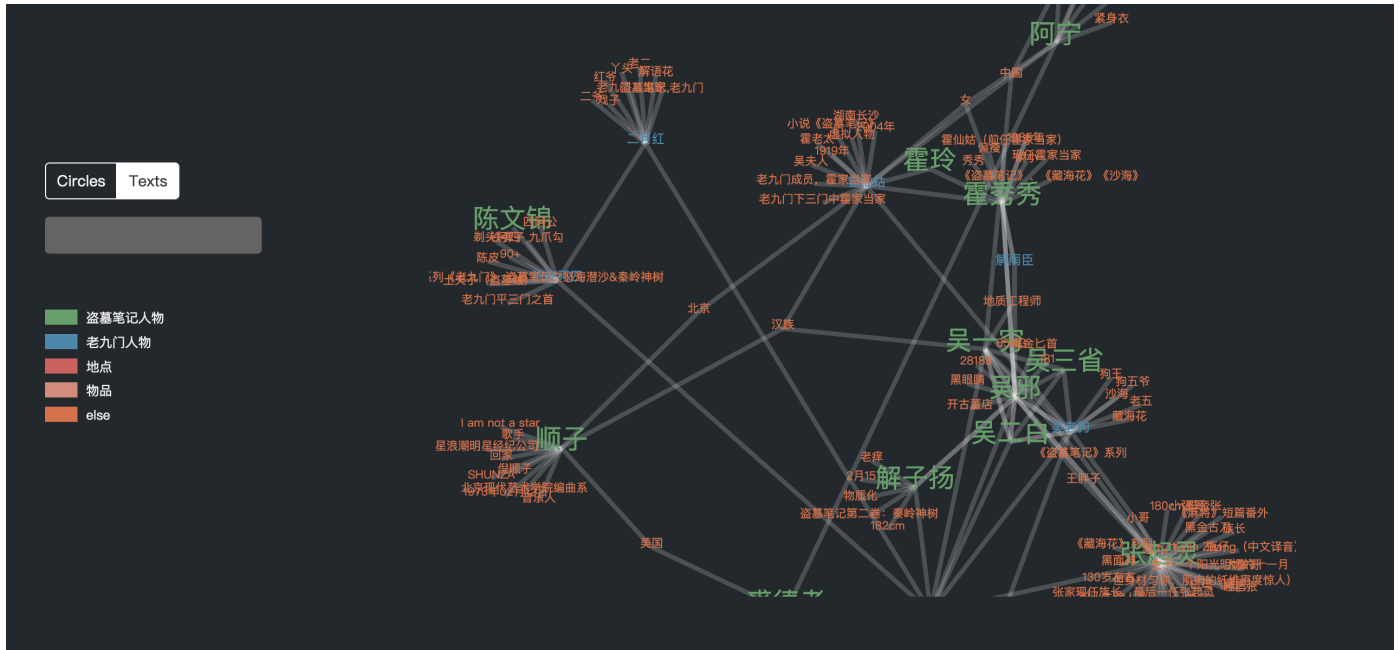
# 获取所有实体的列表
entity_list = []
for node in data["nodes"]:
    entity_list.append(node["id"])

# 将无法在实体集中找到对应的目标实体归类为第五类：else
for link in data["links"]:
    if link["target"] not in entity_list:
        data["nodes"].append(
            {
                'class': 'else',
                'id': link["target"],
                'group': "4",
                'size': "10",
            }
        )

# 更新links_and_nodes.json文件
data = json.dumps(data, ensure_ascii=False)
with open('links_and_nodes.json', 'w', encoding='utf-8') as json_file:
    json_file.write(data)
```

通过以上代码，我们将无法在实体集中找到对应的目标实体归类为第五类："else"。这样一来，我们就能够在可视化过程中更清晰地呈现实体之间的关系，提高了可视化效果和用户体验。

可视化效果如下：



5 问答系统构建

这智能问答系统的设计目标是通过自然语言处理技术和知识图谱数据提供准确的答案。系统主要由数据准备、实体与关系识别、答案生成和用户交互等部分组成。

在数据准备阶段，系统加载实体属性数据和知识图谱数据。实体属性数据包含了每个实体的具体属性信息，而知识图谱数据则描述了实体之间的关系。

在实体与关系识别阶段，系统利用jieba分词工具进行中文分词，并结合词性标注识别问题中的实体和关系。这些识别出的实体和关系被用作后续查询的关键信息。

在答案生成阶段，系统根据识别出的实体和关系，在知识图谱中查询相应的信息，并生成答案。如果是查询实体的属性，则在实体属性数据中查找对应的属性值；如果是查询实体之间的关系，则在知识图谱数据中查找相应的关系信息。

```
import json
import jieba
import jieba.posseg as pseg

# 加载实体属性数据
with open('attr.json', 'r', encoding='utf-8') as f:
    attr_data = json.load(f)

# 加载知识图谱数据
with open('links_and_nodes.json', 'r', encoding='utf-8') as f:
    graph_data = json.load(f)

def query_entity_property(entity, property_name):
    if entity in attr_data and property_name in attr_data[entity]:
        return attr_data[entity][property_name]
    else:
```

```

        return None

def query_relation(source, target):
    for link in graph_data['links']:
        if link['source'] == source and link['target'] == target:
            return link['relation']
    return None

def extract_entities_and_relation(question):
    words = list(pseg.cut(question))
    entities = []
    relation = None

    for word, flag in words:
        if flag.startswith('n'):
            entities.append(word)
        elif flag == 'c':
            relation = word

    return entities, relation

def answer_question(question):
    entities, relation = extract_entities_and_relation(question)
    print(entities, relation)
    if len(entities) < 2:
        return "请提供足够的实体信息来回答问题。"

    if relation:
        source, target = entities[:2]
        relation = query_relation(source, target)
        if relation:
            return f"{source}和{target}之间的关系是{relation}"
        else:
            return f"{source}和{target}之间没有关系"
    else:
        entity = entities[0]
        property_name = entities[1]
        property_value = query_entity_property(entity, property_name)
        if property_value:
            return f"{entity}的{property_name}是{property_value}"
        else:
            return f"{entity}没有{property_name}这个属性"

if __name__ == "__main__":
    while True:

```

```

question = input("请输入您的问题：")
if question.lower() == 'exit':
    print("感谢使用，再见！")
    break
else:
    answer = answer_question(question)
    print(answer)

```

用户可以通过命令行输入问题，系统会根据问题内容生成相应的答案。以下是一个示例：

```

请输入您的问题：吴邪的职业是什么？
吴邪的职业是开古董店

```

通过上述交互，用户能够轻松获取到他们所需要的信息。这个智能问答系统具有一定的智能化和灵活性，能够满足用户的需求，并有望在未来得到进一步的优化和拓展。

```

请输入您的问题：张起灵和吴邪的关系是什么？
['张起灵', '吴邪', '关系'] 和
张起灵和吴邪之间的关系是队友
请输入您的问题：吴邪的职业是什么？
['吴邪', '职业'] None
吴邪的职业是['开古董店']

```

6 总结

通过构建《盗墓笔记》知识图谱及相应的智能问答系统，我们实现了对小说中人物、地点、物品等信息的整合和查询。该系统不仅可以提供具体实体的属性信息，还能回答关于实体之间关系的问题。这为读者提供了一个便捷的方式来获取《盗墓笔记》相关的知识，同时也为后续研究和分析提供了数据基础。未来，我们将继续优化系统的性能，拓展知识图谱的覆盖范围，以提供更加全面和准确的信息服务。