目录

1.摘要

在本次课程设计中我们小组对金庸小说中人物进行知识图谱构建（本体构建，事实抽取，类别推断）。其中，类别推断部分部分我们构建了一个推理机并在此基础上加入了不一致性检测的功能。

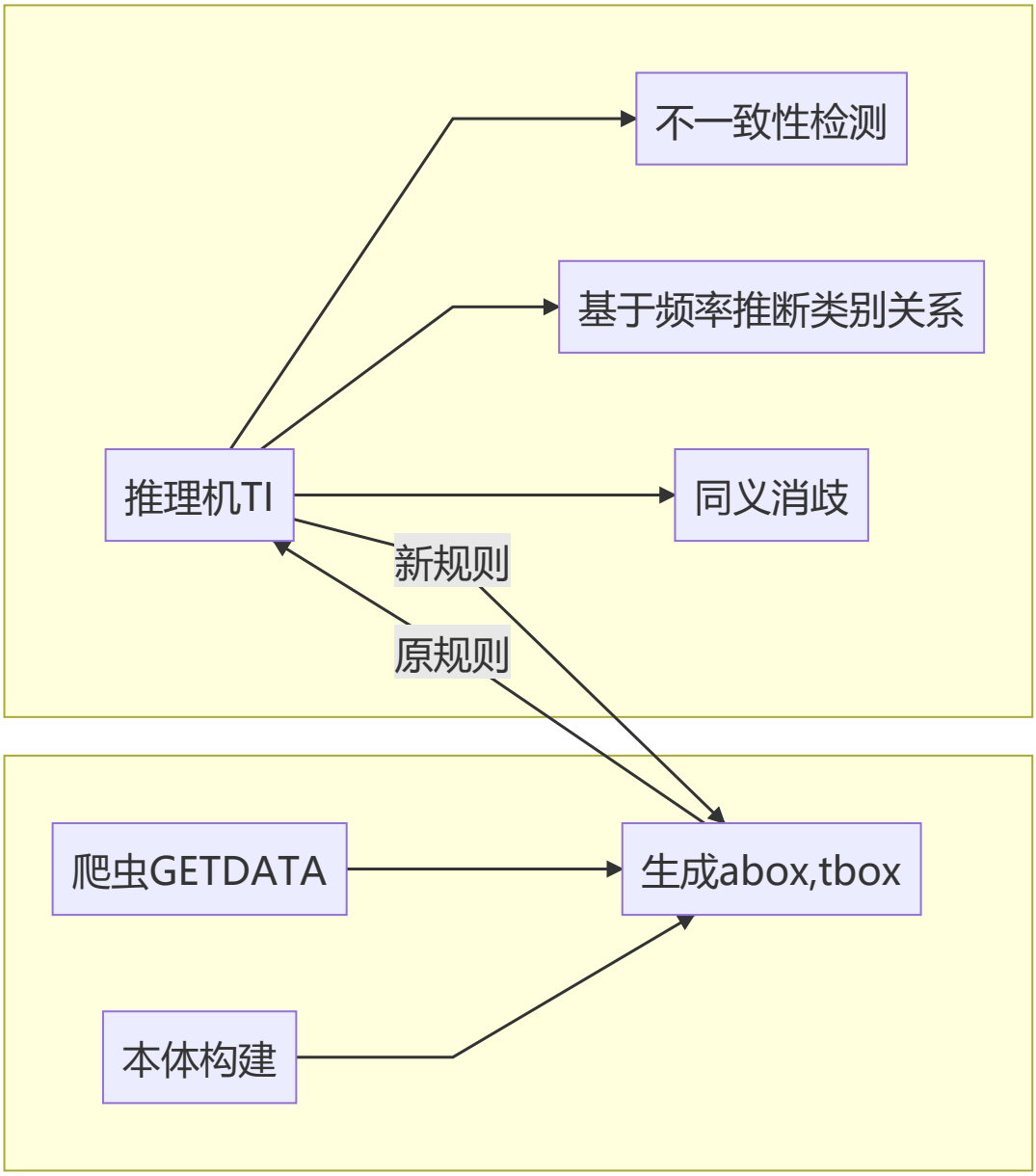
在后面的章节中，第2节中描述了如何运行测试代码并提供了知识图谱的基本信息（例如规模）;第3节介绍了所提交代码的代码结构和各个模块的功能；第4节介绍了各个模块实现的具体细节第5节介绍了我们小组提出的创新想法；第6节描述了小组每人的工作量，第7节为引用部分

2.结果展示与知识图谱基本信息

文件：result/金庸小说人物知识图谱.rdf
注明：result文件夹内其他文件为生成知识图谱的过程文件

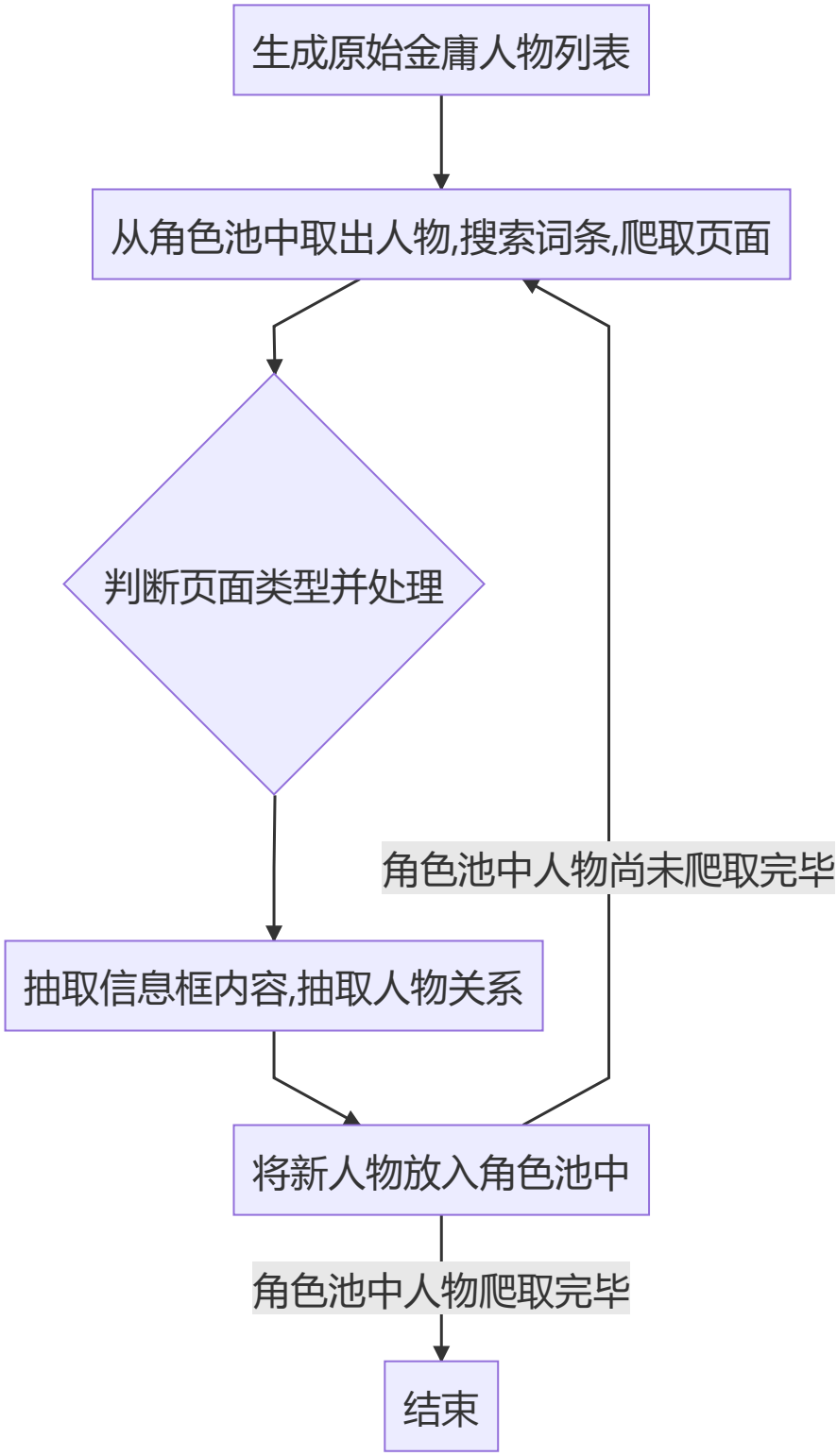
规模：3508个实体。abox规则18366条；tbox规则6687条;节点数634个(金庸小说人物数量)
注明：知识图谱实际规模与答辩时所述不一致。规模大小的检查可以使用‘金庸小说人物知识图谱.rdf’，节点数检查可以使用’abox_test_1.csv’

3.实现思路与代码组织

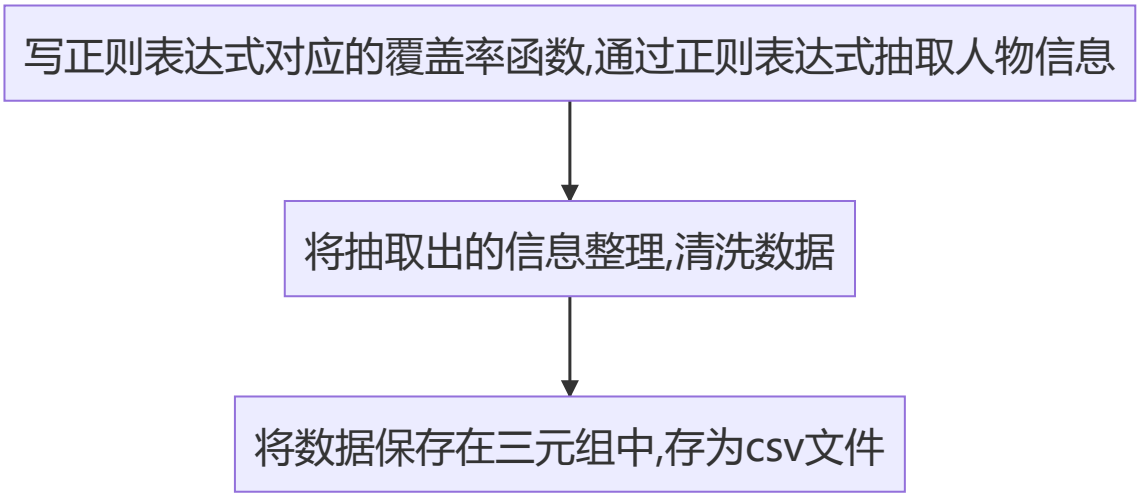


爬虫和事实抽取封装为库GETDATA;推理机封装为库TI;本节后面的内容首先介绍GETDATA的代码结构和执行流程，再介绍TI的代码结构和执行流程

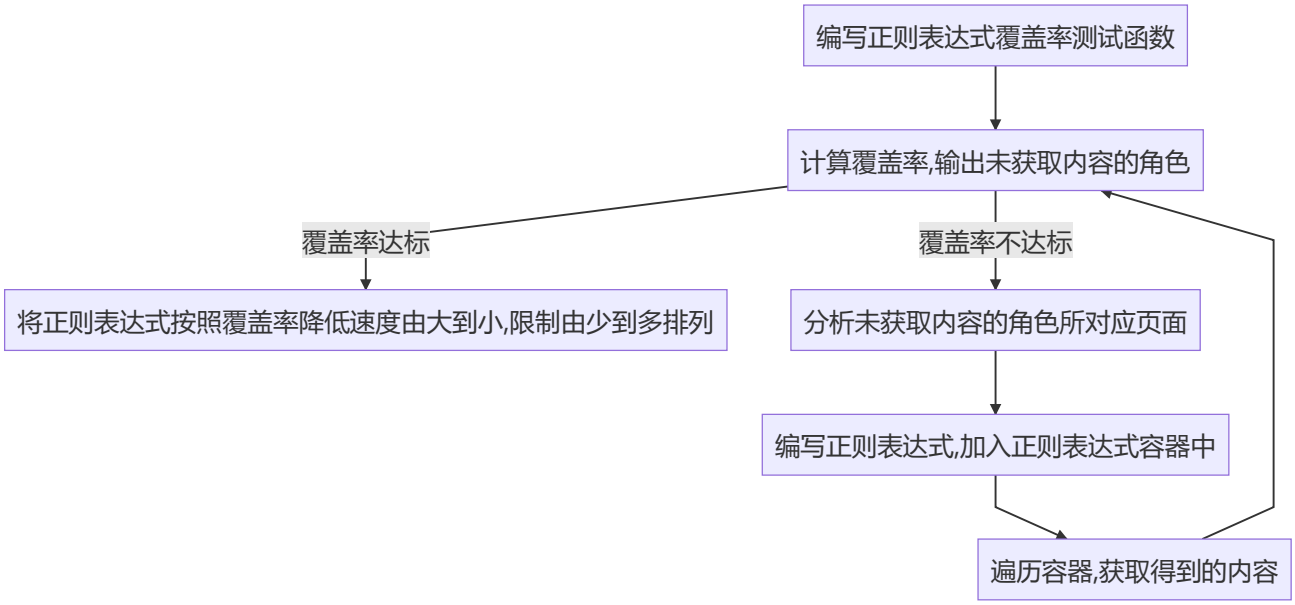
其中爬虫流程如下



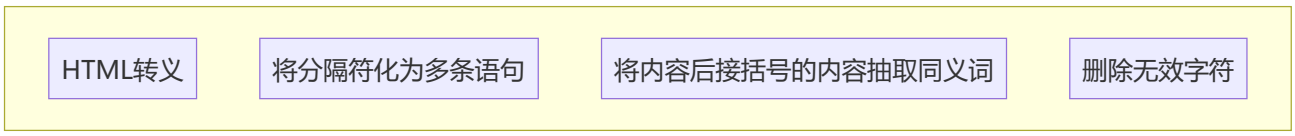
事实抽取部分使用正则表达式抽取信息框内容,流程如下



写正则表达式抽取信息过程中,采用如下方法，开发效率较高。



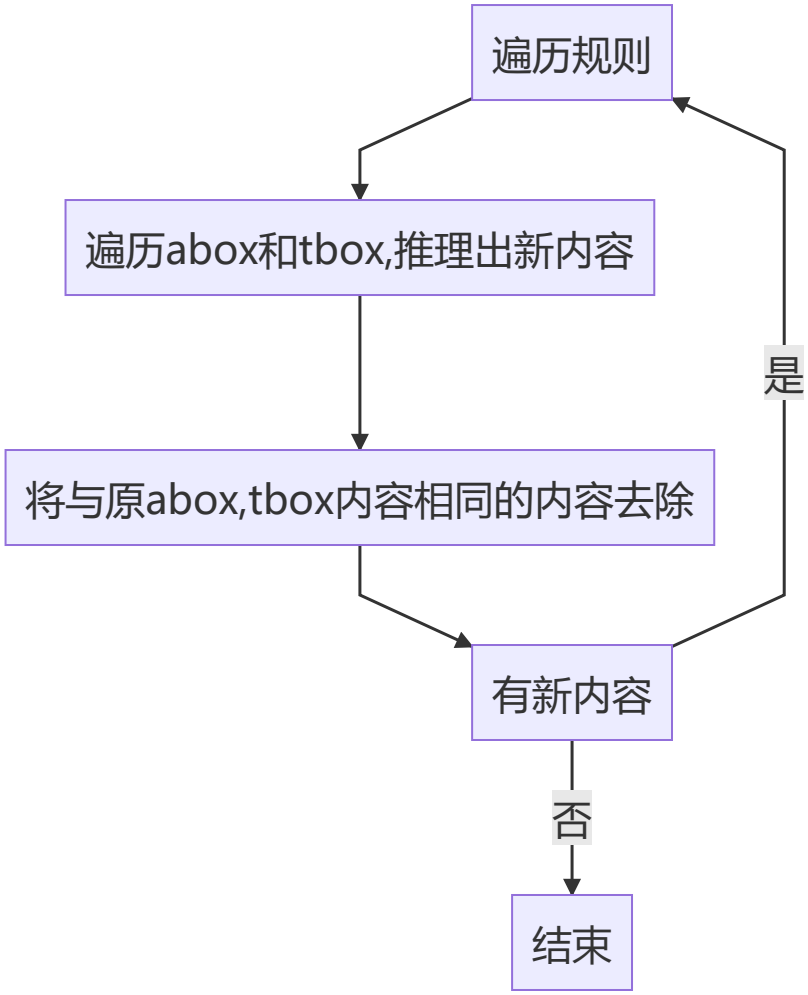
清理数据方面我们小组完成的工作如下



我们小组实现的推理机包含如下规则：(规则内容请见第6节)

- rdf和rdfs规则
- 部分owl规则
- 基于频率推断subclassof 和subproperty的规则
- 不一致性检测规则
- 生成sameas用于同义消歧

推理机运行流程如下



时间复杂度：由于能够触发规则的事实稀疏，所以推理过程的时间复杂度没有原预想的那样高n^{x+3},在可以接受的范围($\approx n^2$)

本节最后介绍不一致性检测和同义消歧实现过程。

1.不一致性检测使用的是6.4节所述规则。6.4节的规则是针对实例和类的，但是实际上也包含了类与类的不一致性检测，如:

$$\frac{A \perp B \wedge CsubclassofA \wedge CsubclassofB}{C = \emptyset} \tag{1}$$

2.同义消歧
2.1首先使用基于频率的方法和functional,inverse_functional的规则会推导出某些实例，类同义。
2.2利用同义信息尽可能扩大数据集，以将尽可能多地同义信息找出
2.3将所有同义词关联到一个实体上完成同义消歧（TI.cleansame）

4.代码细节

第4节主要介绍GETDATA中characterlist_base，LouisCha_characterlist；TI部分的rule_basic和typeinference的实现。

4.1GETDATA

由于我们小组在设计时采用了模板方法，将获取X信息的任务抽出共性部分，封装为基类，使得在抽取其他类似信息（最简单的如抽取水浒传人物信息，可以通过重载预设的函数来玩完成诸如抽取公司高管信息的工作）

以下为基类代码。第3节中所提到的爬虫流程的代码由‘run’函数完成,run函数将会调用继承类实现的函数来完成对具体任务的流程

```
class characterlist_base():
    '''
    这个类用来继承，必须实现：
    generate_url(self,args)
    get_relative_character(self,args)
    '''
    def start(self):
        self.pointer=0
        self.namedict=dict()
        self.namelist=[]
        self.validnamelist=[]
    def _init_namelist_dict(self,namelistroot):
        '''
        按照文本内容返回最原始的人物列表。
        **最后的去重操作会将顺序打乱，所以不可重复操作**
        :param namelistroot: 储存人物列表的json文件地址
        json文件要求：键存储书名，值存储列表，列表内为人名，人名以字符串给出
        :return:无                          #TODO:True为读取文件成功，False为失败
        '''
        with open(namelistroot,'r',encoding='utf-8')as f:
            self.namedict=eval(f.read())
            for value in self.namedict.values():
                self.namelist.extend(value)
        self.namelist=list(set(self.namelist))
        print('将{} 作为人物列表导入完成'.format(namelistroot))
    def init_all(self,namelistroot):
        self.pointer=0
        self._init_namelist_dict(namelistroot)
        self.validnamelist = []
    def generate_url(self,args=dict()):
```

```
        raise NotImplementedError()
    def findbookname(self,charactername):
        for key,value in self.namedict.items():
            if charactername in value:
                return key
        return ''
    def deal_no_jump(self,args:dict):
        raise NotImplementedError()
    def get_relative_character(self,args=dict()):
        raise NotImplementedError()
    def cleannames(self):
        raise NotImplementedError()
    def run(self,args:dict):
        '''
        模板方法,获得所有人物的百科html文件
        saveroot,host_url,namelistroot
        :param args:
        :return:
        '''
        self.start()
        self.init_all(args['namelistroot'])
        while True:
            if self.pointer>=len(self.namelist):
               break
            else:
                #输出前先清空输出
                os.system('cls')
                print('已经爬完{},当前进度<=
{:.2%}'.format(self.pointer,self.pointer/len(self.namelist)))
            url=self.generate_url(args)
            rawdata=get_savehtml(url,args['saveroot'],self.namelist[self.pointer])
            if rawdata!="":#爬取出了结果
                rawdata=self.deal_no_jump(args={
                    'data':rawdata,
                    'charactername':self.namelist[self.pointer],
                    'saveroot':args['saveroot'],
                    'host_url':args['host_url']
                })
                if rawdata==-1:
                    self.pointer+=1
                    continue
                self.validnamelist.append(self.namelist[self.pointer])#找到哪些人名有词条
                names=self.get_relative_character({'data':rawdata})
                names=self.cleannames({'dirty_characters':names})
                for name in names:
                    if name not in self.namelist:
                        self.namelist.append(name)
            self.pointer += 1
            time.sleep(rand.randint(1,10)/10)
    def __str__(self):
        return str(
            dict(
                {
                    'pointer':self.pointer,
                    'namelist':self.namelist,
                    'namedict':self.namedict,
                    'validnamelist':self.validnamelist
                }
            )
        )

```

以下为继承类代码，用于实现本次课设的“金庸小说人物”。

```
class LouisCha_characterlist(characterlist_base):
    def generate_url(self, args):
        return
f'https://baike.baidu.com/item/{tranchinise(self.namelist[self.pointer])}'
```

```
    def get_relative_character(self, args):
        '''#获取人名（脏数据）
        df = pd.read_html(args['data'])[0]
        return list(df['人名'])'''#不够直接
        pat='''<li class="lemma-relation-item">.*?<span class="title">(.*?)</span>.*?</li>'''#抽人物关系
        return re.compile(pat,re.S).findall(args['data'])
    def is_clean(self,string):
        return string.find(':') == -1 and string.find('（') == -1 and\
        string.find('）') == -1 and string.find('  ')==-1 and\
        string.find('、')==-1
    def cleannames(self,args:dict):
        # TODO:清理数据，样例：['父亲：黄药师  母亲：冯氏（小字阿衡）', '郭靖', '郭芙、郭襄、郭破虏', '洪七公', '郭啸天', '李萍', '杨过', '耶律齐', '穆念慈、周伯通', '双雕、小红马、血鸟（连载版）']
        dirty_characters=args['dirty_characters']
        clean_characters=[]
        for dirty_character in dirty_characters:
           #思路是把连续的中文挑出来，如果挑出非人名也无妨，因为deal no jump会将其去掉
            tran=re.compile('([\u4e00-\u9fa5]+)').findall(dirty_character)
            clean_characters.extend(tran)
        return clean_characters
    def _nobookappearance(self,args):
        for elem in self.namedict.keys():
            if args['data'].find(elem)!=-1:
                return False
        return True
    def deal_no_jump(self,args:dict):
        '''
                这部分较为困难。
                没有找到词条也在这里处理
                :param args:
                data
                charactername
                saveroot
                host_url
                :return:
                '''
        if args['data'].find('金庸') == -1 and self._nobookappearance(args):  # 如‘冯氏’，搜出来不是金庸小说人物
            '''
            此处不能只写含"金庸"，如"谭婆：小说《天龙八部》中的人物"
            '''
            return -1
        elif args['data'].find('keywords') == -1:
            '''
            #找字串，-1为对象字符串不含参数字符串，其他值为索引
            #百科搜索可能页面跳转，找规律发现未跳转的页面没'keyword'字符串
            '''
            # TODO:通过返回码判断是否跳转更安全
            '''
            找规律发现金庸人物的目录下含书名,用人物找书名
            '''
            if args['data'].find('百度百科尚未收录词条') != -1:
                return -1

            else:
                newurl = ''
                polysemous_words = strongpat(args['data'], {
                    'for_polysemous': '<div class="para" label-module="para"><a(.*?)</a>[\s]*</div>'  # 识别多义词,把链接也拉过来
                }, re.S)

                for elem in polysemous_words[0]:
                    if elem.find('金庸') != -1 or elem.find(
                            self.findbookname(args['charactername'])) != -1:  # 认为结果含有金庸 或是书名就是索要查找结果
                        tran = re.compile('href="(.*?)"').findall(elem)  # 找链接
                        newurl = args['host_url'] + tran[0][1:]
```

```
                        return get_savehtml(newurl, args['saveroot'],
args['charactername'])

        return args['data']
    def _getinfobox(self,args:dict):
        '''
        百度百科的html文档中infobox有多种格式,需要多种正则表达式进行提取。本函数提供了一个比较有效的
方法。
        这一部分是这样完成的:
        1.首先创建测试函数，测试函数会将正则表达式提取为空的文档输出，并输出提取失败的文档个数
        2.根据提取失败的文档写出新的正则表达式添加在pats里
        3.本函数会循环迭代pats，使得文档尽可能多地匹配
        4.注意将较偏的提取规则放在列表的后面，以防其他html使用较偏的提取规则提取，如公孙绿萼.html的
infobox不够规范，只能使用如pats[4]的提取规则提取
        :param args:
        :return:
        '''
        pats=['''<dt class="basicInfo-item name">(.*?)</dt>[\s]*<dd class="basicInfo-
item value">[\s]*(.*?)[\s]*</dd>''',
              '''<li class="basicInfo-hide">[\s]*<div class="info-title">(.*?)</div>
[\s]*<div class="info-content".*?>(.*?)</div>[\s]*</li>''',
              '''<li>[\s]*<div class="info-title">(.*?)</div>[\s]*<div class="info-
content".*?>(.*?)</div>[\s]*</li>''',
              '''<p>[\s]*<strong>(.*?)</strong>(.*?)</p>''',
              '''\n([\u2E80-\u9FFF]*?)：(.*?)<br />''']

        for pat in pats:
            tran=re.compile(pat,re.S).findall(args['data'])
            if tran!=[]:
                rst=tran[:]
                return [(args['name'],rst[i][0],rst[i][1]) for i in range(len(rst))]
    def test_getinfobox(self,args,key):
        count=0
        for elem in self.getzip(args):
            if self._getinfobox(elem)==None:
                count+=1
                print(elem['name'])
        print(count)
    def getzip(self,args:dict):
        for elem in os.listdir(args['rawdataroot']):
            yield {'name':elem[:-5],
                   'data':open(args['rawdataroot']+elem,'r',encoding='utf-8').read()}
    def get_triple(self,args:dict):
        import zipfile
        zipfile_path = ''
        trandict=geti2c()
        with zipfile.ZipFile(args['rawdataroot'], mode='r') as zfile:  # 只读方式打开压缩
包
            for name in zfile.namelist():  # 获取zip文档内所有文件的名称列表
                with zfile.open(name, mode='r') as single_file:
                    yield self._getinfobox(args=
{'name':trandict[name[:-5]],'data':single_file.read()})
    def getinfobox(self,args):
        target=open('../test.csv', 'a', encoding='utf-8')
        writer=csv.writer(target)
        for newargs in self.getzip(args):
            tran=self._getinfobox(args=newargs)
            if tran:
                for triple in tran:
                    writer.writerow(triple)
        target.close()
```

4.2 TI

推理机TI整体架构为由typeinference所提供的功能循环调用不同的规则类(rule)，所有的规则继承自Rule_basic统一封装接口。

以下为Rule_basic具体实现

```
class Rule_basic():
    def _run(self, args):
        '''
        接口说明：
        0.类命名规则：rule数字，数字对应ppt中rdf后面数字
        派生类请单独写在一个文件中，本文件下的类作为例子。
        1.在本函数中需要完成的工作：根据给定三元组组成的列表（abox，tbox），按照当前函数所执行的规则编写输出两个新三元组组成的列表的接口。两个新三
        元组分别为abox和tbox。新三元组中不包含原三元组列表的内容。不需要去重。
        2.本函数执行的操作是按照当前规则，将生成的三元组全部输出
        :param args: [
        参数规定：
        'tbox':tbox,
        'abox':abox
        可以按照 rule数字_参数名 添加新键，命名规则保证不重名
        ]
        trule[0]:头，trule[1]:边，trule[2]:尾
        arule[0]:头，arule[1]:边，arule[2]:尾

        迭代器命名建议：trule对应tbox的迭代器，arule对应abox的迭代器。

        :return:返回值建议命名为rst_abox,rst_tbox

        type 省略前缀
        3.TODO:本函数所完成的逻辑规则(需填写)
        '''
        raise NotImplementedError()

    def run(self, args):
        raw_rst_abox, raw_rst_tbox = self._run(args)
        rst_abox, rst_tbox = [], []
        for elem in raw_rst_abox:
            if elem not in args['abox']:
                rst_abox.append(elem)
        for elem in raw_rst_tbox:
            if elem not in args['tbox']:
                rst_tbox.append(elem)
        return rst_abox, rst_tbox
```

以下为typeinference的具体实现

```
class typeInference():
    def start(self,rules):
        self.IE=rules
        pass
    def retranslate(self):
        pass
    def TI(self,args):
        rst_abox,rst_tbox=args['abox'],args['tbox']
        counter=1
        import time
        timer=[]
        curtime=time.time()
        while(True):
            new_abox,new_tbox=[],[]
            for rule in self.IE :
                tran=rule.run(args={
```

```
                    'abox':rst_abox,
                    'tbox':rst_tbox
                })
                new_abox.extend(tran[0])
                new_tbox.extend(tran[1])

            if len(new_abox)==0 and len(new_tbox)==0:
                break
            rst_abox.extend(new_abox)
            rst_tbox.extend(new_tbox)
            #外层去重
            rst_abox=list(set([tuple(elem) for elem in rst_abox]))
            rst_tbox = list(set([tuple(elem) for elem in rst_tbox]))
            tran=time.time()-curtime
            timer.append(tran)
            print('第{}次迭代,消耗时间:{:.2f}s'.format(counter,tran))
            counter+=1
        print('总耗时:{:.2f}s'.format(sum(timer)))
        return rst_abox, rst_tbox

```

不一致性检测代码如下

```

class rule_check_1(Rule_check_basic):
    '''对于类A和类B,若A和B不相交且存在a使得a属于A且a属于B,
    则返回一条标识错误的三元组('序列号','DISJOINTERROR','ainA_ainB')
    rule_check规则禁止改动规则
    '''
    def _run_tran(self):
        rst_assertion_box= []
        rst_original_box=self.rst_original_box[:]
        #找所有含type的语句
        tran1,tran2=[],[]
        for elem in rst_original_box:
            if elem[1]==TI_TYPE:
                tran1.append(elem)
            if elem[1]==TI_DISJOINT:
                tran2.append(elem)
        #分析type语句
        for elem1 in tran2:
            for elem2 in tran1:
                if elem2[2]==elem1[0]:
                    for elem3 in tran1:
                        if elem3[0]==elem2[0] and elem3[2]==elem1[2]:
                            errorsrcipt=f'{elem3[0]} in {elem1[0]} and {elem1[2]},
{elem1[0]} disjointwith {elem1[2]}'
                            rst_assertion_box.extend(
                                [
                                    (elem1[3],TI_DISJOINT_ERROR,errorsrcipt),
                                    (elem2[3], TI_DISJOINT_ERROR, errorsrcipt),
                                    (elem3[3], TI_DISJOINT_ERROR, errorsrcipt),
                                ]
                            )
        return  rst_original_box,rst_assertion_box
```

同义消歧代码如下:

```
def cleansame(abox,tbox):#完成同义消歧
    for i,arule in enumerate(abox):
        for j,elem1 in enumerate(arule):
            if elem1==TI_SAMEAS:
                for i1,arule1 in enumerate(abox):
                    for j1, elem11 in enumerate(arule1):
                        if elem11 == arule[0]:
                            arule1[j1]=arule[2]
```

```
                for i1, trule1 in enumerate(tbox):
                    for j1, elem11 in enumerate(trule1):
                        if elem11 == arule[0]:
                            trule1[j1] = arule[2]
    #删除重复词，原来的sameas用于为实体添加别名
    rst_abox=list(set(abox))
    rst_tbox = list(set(tbox))
    return rst_abox,rst_tbox
```

5.创新点

任务：使用对抗文本抽取属性的值（属性值的合并）
在文本中抽取属性值时经常会出现某个属性的值有多种表达方式(质量:高 可以表达成 ‘优秀’、‘高质量’)，现在需要将这些属性值找出或整合（消歧）。
给定一个至少是闭源的分类模型(可以获取模型的输入输出，但不知道模型的具体结构)。

算法：

这个方法的思想是认为经过对抗训练，且有较好分类效果的分类器可以逼近人的分类标准。构建可以区分属性值的分类器后，再通过计算后验概率或梯度的方法在分类器起较大作用的单词，将此作为属性的同义词。

1. 选取多个经过对抗训练的分类器进行如下操作，每个分类器分类的是某个属性的值（如每个分类器可以分类文本中表达的产品质量信息，分类为‘好’，‘中’，‘差’）
2. 计算分类器中占较大贡献度的单词
3. 将3中得到的单词放入一个属性值的候选池中，计数
4. 单词对应的候选词投票，得到与属性值同义的词

其中步骤2.参考了[1],计算每个词在分类中所占的贡献度

这个方法是使用如下函数

$$C_F(a_i,y)=\begin{cases} p_F(y_i|s)-p_F(y_i|s\ without\ w_k) & F(s)=y_i=F(s\ without\ w_k) \\ p_F(y_i|s)+p_F(y_i|s\ without\ w_k) & F(s)=y_i\neq F(s\ without\ w_k) \end{cases} \quad (2)$$

其中 P_F是分类器F所认为的某文本s(或s without w_k)为y_i类别的概率。将分类器看作一个以文本为输入类别为输出的函数F,通过上式可以计算每个词的贡献度。

在模型完全已知(知道模型损失函数)的情况下，可以使用类似FGSM的方法来计算贡献度。

$$C_F(w_k,y)=-\nabla_{w_k}J(F,s,y_i) \quad (3)$$

其中J为模型的损失函数。

6.实现规则

本次课程设计所实现的规则:

6.1rdf,rdfs规则

$$\frac{xRy}{R\,type\,Property} \tag{4}$$

$$\frac{R\,domain\,C_1 \wedge\ xRy}{x\,type\,C_1} \tag{5}$$

$$\frac{R\,range\,C_1 \wedge\ xRy}{y\,type\,C_1} \tag{6}$$

$$\frac{xRy}{x\,type\,Resource \wedge y\,type\,Resource} \tag{7}$$

$$\frac{R_1\ subpropertyof\ R_2 \wedge R_2\ subpropertyof\ R_3}{R_1\ subpropertyof\ R_3} \tag{8}$$

$$\frac{R\,type\,Property}{R\,subpropertyof\,R} \tag{9}$$

$$\frac{R_1\ subpropertyof\ R_2 \wedge xR_1y}{xR_2y}\ \frac{R_1\ subpropertyof\ R_2 \wedge xR_1y}{xR_2y} \tag{10}$$

$$\frac{c\,type\,Class}{c\,sublassof\,Resource} \tag{11}$$

$$\frac{c_1\ sublcassof\ c_2 \wedge x\,type\,c_1}{x\,type\,c_2} \tag{12}$$

$$\frac{c\,type\,Class}{c\,subclassof\,c} \tag{13}$$

$$\frac{c_1\ subclassof\ c_2 \wedge c_2\ subclassof\ c_3}{c1\ subclassof\ c_3} \tag{14}$$

$$\frac{x\,type\,Datatype}{x\ subclassof\ Literal} \tag{15}$$

6.2部分owl规则

$$\frac{R\,type\,Transitive \wedge xRy \wedge yRz}{xRz} \tag{16}$$

$$\frac{R\,type\,Symmetric \wedge xRy}{yRx} \tag{17}$$

$$\frac{R\,type\,Functional \wedge xRy_1 \wedge xRy_2}{y_1\ sameas\ y_2} \tag{18}$$

$$\frac{x_1\ sameas\ x_2 \wedge f(x_1)}{f(x_2)} \tag{19}$$

$$\frac{R_1\ inverseof\ R_2 \wedge xR_1y}{yR_2x} \tag{20}$$

$$\frac{R\,type\,InverseFunctional \wedge x_1 R y \wedge x_2 R y}{x_1\ sameas\ x_2} \tag{21}$$

6.3基于频率推断subclassof和subpropertyof

$$\begin{gathered} R_i = \{< x_{i1}, y_{i1} >, \ldots, < x_{in}, y_{in} >\} \\ R_j = \{< x_{j1}, y_{j1} >, \ldots, < x_{jn}, y_{jn} >\} \\ \frac{len(R_i \cap R_j) \div len(R_i) > \alpha}{R_i \subseteq R_j} \end{gathered} \tag{22}$$

6.4不一致性(得到Error同时会输出错误信息)

$$\frac{A \perp B \wedge a\,type\,A \wedge a\,type\,B}{Error} \tag{23}$$

6.5生成sameas用于同义消歧

$$\frac{c_1\ subclassof\ c_2 \wedge c_2\ subclassof\ c_1}{c1\ sameas\ c2} \tag{24}$$

$$\frac{p_1\ subpropertyof\ p_2 \wedge p_2\ subpropertyof\ p_1}{p1\ sameas\ p_2} \tag{25}$$

6.小组每人工作量

略

7.引用

[1] Samanta, S., & Mehta, S. (2017). *Towards Crafting Text Adversarial Samples.*