信息抽取实验系统作业

2018211517 何泓川
2018211512 鲁嘉祺

实验目的

设计实现一个信息抽取实验系统，，特定领域语料根据自己的兴趣选定，规模不低于 100 篇文档，进行本地存储。对自己感兴趣的特定信息点进行抽取，并将结果展示出来。其中，特定信息点的个数不低于 5 个。可以调用开源的中英文自然语言处理基本模块，如分句、分词、命名实体识别、句法分析。信息抽取算法可以根据自己的兴趣选择，至少实现正则表达式匹配算法的特定信息点抽取。最好能对抽取结果的准确率进行人工评价。

实验环境

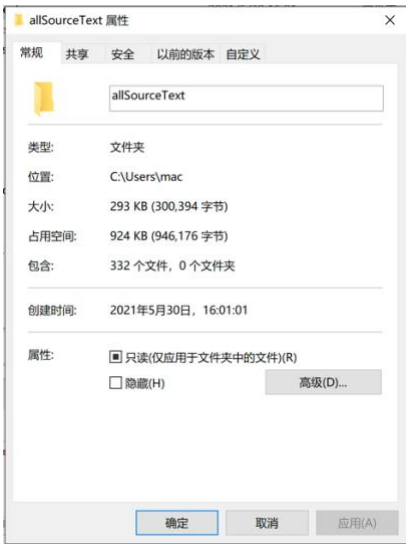
python3.9

完成情况

本实验完成了 1~5 全部要求。

实验过程

（1） 数据集获取
本程序语料库选择上海大学（语义智能实验室）所构建的 CEC 语料库，获取地址为 https://github.com/shijiebei2009/CEC-Corpus/tree/master/raw%20corpus/allSourceText，本程序使用的工具为 stanfordcorenlp。该语料库共有 332 篇文档，如下图：



（2） 信息抽取过程

对于一篇文章，先调用 StanfordNLP 函数，获得文章进行分词、标注词性后的结果。再创建一个 infoExtraction 对象，再对文章进行信息抽取。调用成员函数 extractEvent 抽取事件类型。对所有类型设置触发词，统计文章中哪一类的触发词出现最多，事件的类型即为这一类，以及这一类中哪个触发词出现最多，事件的描述为根据这一触发词来描述，并把这一触发词作为对象的触发词，如下图：

```
def extractEvent(self):
    fire_trigger_list = ['森林', '爆燃', '火灾', '起火', '泄漏']
    quake_trigger_list = ['地震', '震感', '摇晃']
    traff_trigger_list = ['车', '中巴', '水泥罐车', '辆', '相撞', '撞上', '开车', '交通事故']
    terro_trigger_list = ['恐怖', '武装', '炸弹', '自杀', '爆炸']
    food_trigger_list= ['中毒', '食物', '呕吐', '恶心']
    cnt = dict()
    cnt['fire'] = 0
    cnt['qua'] = 0
    cnt['traf'] = 0
    cnt['terro'] = 0
    cnt['food'] = 0
    fire_set = dict()
    quake_set = dict()
    traff_set = dict()
    terro_set = dict()
    food_set = dict()
    for item in self.nlp_result:
        if item[0] in fire_trigger_list:
            cnt['fire'] += 1
            if item[0] not in fire_set:
                fire_set[item[0]] = 1
            else:
                fire_set[item[0]] += 1
        elif item[0] in quake_trigger_list:
            cnt['qua'] += 1
            if item[0] not in quake_set:
                quake_set[item[0]] = 1
            else:
                quake_set[item[0]] += 1
        elif item[0] in traff_trigger_list:
            cnt['traf'] += 1
            if item[0] not in traff_set:
                traff_set[item[0]] = 1
            else:
                traff_set[item[0]] += 1
```

```
        elif item[0] in terro_trigger_list:
            cnt['terro'] += 1
            if item[0] not in terro_set:
                terro_set[item[0]] = 1
            else:
                terro_set[item[0]] += 1
        elif item[0] in food_trigger_list:
            cnt['food'] += 1
            if item[0] not in food_set:
                food_set[item[0]] = 1
            else:
                food_set[item[0]] += 1

    kind = max(cnt, key=cnt.get)
    if kind == 'fire':
        trigger = max(fire_set, key=fire_set.get)
        self.event['trigger'] = trigger
        self.event['events'] = settings.FIRE_TRIGGER[trigger]
        return
    if kind == 'qua':
        trigger = max(quake_set, key=quake_set.get)
        self.event['trigger'] = trigger
        self.event['events'] = settings.EAR_QUAKE_TRIGGER[trigger]
        return
    if kind == 'traf':
        trigger = max(traff_set, key=traff_set.get)
        self.event['trigger'] = trigger
        self.event['events'] = settings.TRAFF_TRIGGER[trigger]
        return
    if kind == 'terro':
        trigger = max(terro_set, key=terro_set.get)
        self.event['trigger'] = trigger
        self.event['events'] = settings.TERRO_TRIGGER[trigger]
        return
    if kind == 'food':
        trigger = max(food_set, key=food_set.get)
```

```
        self.event['trigger'] = trigger
        self.event['events'] = settings.FOOD_TRIGGER[trigger]
        return

    self.event['events'] = None      BruceHo202, 5 days ago • Add files via upload
    self.event['trigger'] = None
```

有了事件类型以后提取相关的时间、地点、相关组织。时间提取方法为根据标注的词性，提取出连续的 DATE、TIME、NUMBER、MISC 的词组成一个时间，返回文章中所有这样时间 list。提取地点的方法为根据标注的词性，提取出连续的 LOCATION, FACILITY, CITY, GPE 的词组成一个地点，返回地点 list。提取相关组织的方法为根据标注词性为 ORGANISATION 的连续的词，返回相关组织的 list。时间的提取方法如下：

```
def extractTime(self):      You, 37 minutes ago • 信息抽取格式
    i = 0
    state = Fa
    time_fire
    result = [
    while i < len(self.nlp_result):
        if self.nlp_result[i][1] in ['DATE', 'TIME']:
            time_fire += self.nlp_result[i][0]
            state = True
        elif self.nlp_result[i][1] in ['NUMBER', 'MISC']:
            time_fire += self.nlp_result[i][0]
        else:
            if state:
                result.append(time_fire)
            time_fire = ""
            state = False
        i += 1
    if state:
        result.append(time_fire)

    return result
```

```
(function) len: (__obj: Sized) -> int
Return the number of items in a container.
```

再根据触发词决定调用哪一个事件函数。本程序选择的语料库有五类事件：火灾、地震、交通事故、恐怖袭击、食品安全。事件函数内部根据该事件的模式匹配到文章的内容，保存到对象里。再根据伤亡信息的模式匹配到伤亡信息，保存到对象里。火灾时间的事件函数如下：

```
def fire_event(self):

    self.cause = pattern_match(pattern_info(FIRE), self.news)
    if len(self.cause) == 0:
        self.cause = "正在进一步调查"
    else:
        self.cause = ",".join(str(i) for i in self.cause)

    self.lose = pattern_match(pattern_lose(), self.news)
    if len(self.lose) == 0:
        self.lose = "未知"
    else:
        self.lose = ",".join(str(i) for i in self.lose)
    self.event['cause'] = self.cause
    self.event['loss'] = self.lose
```

事件及伤亡信息的模式用正则表达式表示，用 re 库进行模式匹配。各类型事件的正则表达式如下：

```
def pattern_info(type):
    if type == FIRE:      BruceHo202, 5 days ago • Add files via upload
        patterns = []
        key_words = ['起火', '事故', '火灾','爆炸','目前']
        pattern = re.compile('.*?(?:{0})原因(.*?)[,.?:;!, 。？：；！]'.format('|'.join(key_words)))
        patterns.append(pattern)
        patterns.append(re.compile(r'(\d+事故因)'))
        patterns.append(re.compile(r'由于.*引发'))

        return patterns
    if type == EAR_Q:
        patterns = []
        pattern = re.compile('[一二三四五六七八九十].[一二三四五六七八九十]级|\\d\\.\\d级')
        patterns.append(pattern)
        return patterns
    if type == TRAFF:
        patterns = []
        pattern = re.compile('(?<=[。，])(?:.*)撞上(?:.*)(?=[。，])')
        patterns.append(pattern)
        pattern = re.compile('(?<=[。，])(?:.*)(?:.*)相[撞碰]')
        patterns.append(pattern)
        return patterns
    if type == TERRO:
        patterns = []
        pattern = re.compile('(?<=[。，])(?:.*)恐怖(?:.*)(?=[。，])')
        patterns.append(pattern)
        pattern = re.compile('(?<=[。，])(?:.*)袭击(?:.*)(?=[。，])')
        patterns.append(pattern)
        pattern = re.compile('(?<=[。，])(?:.*)发生(?:.*)爆炸(?:.*)(?=[。，])')
        patterns.append(pattern)
        return patterns
    if type == FOOD:
        patterns = []
        pattern = re.compile('(?<=[。，])(?:.*)[(食物中毒)(呕吐)](?:.*)(?=[。，])')
        patterns.append(pattern)
        return patterns
```

伤亡模式正则表达式如下：

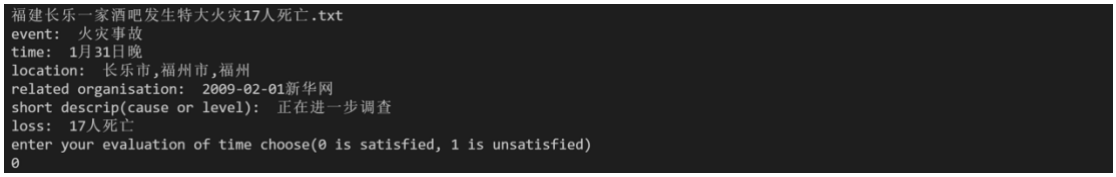
```
def pattern_lose():
    patterns = []

    key_words = ['伤亡', '损失']
    pattern = re.compile('.*?(未造成.*?(?:{0}))[,.?:;!, 。？：；]'.format('|'.join(key_words)))
    patterns.append(pattern)
    patterns.append(re.compile(r'(\d+(?:\w+)死亡)'))
    patterns.append(re.compile(r'(\d+(?:\w+)身亡)'))
    patterns.append(re.compile(r'(\d+(?:\w+)受伤)'))
    patterns.append(re.compile(r'(\d+(?:\w+)烧伤)'))
    patterns.append(re.compile(r'(\d+(?:\w+)坠楼身亡)'))
    patterns.append(re.compile(r'(\d+(?:\w+)遇难)'))
    patterns.append(re.compile(r'(\d+(?:\w+)被灼伤)'))
    patterns.append(re.compile(r'(\d+死)'))
    patterns.append(re.compile(r'(\d+伤)'))
    patterns.append(re.compile(r'[一二三四五六七八九十]+死'))
    patterns.append(re.compile(r'[一二三四五六七八九十]+伤'))
    patterns.append(re.compile(r'\d+(?:\w*)住院'))
    return patterns
```

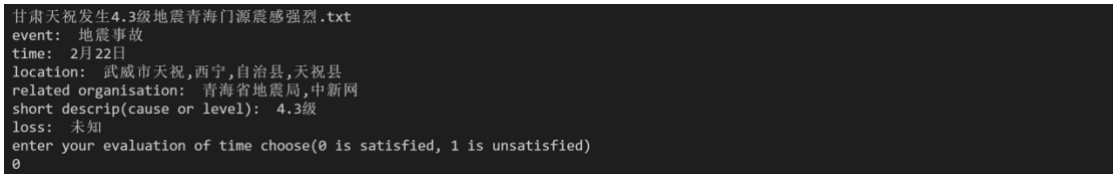
（3） 人工评价准确率

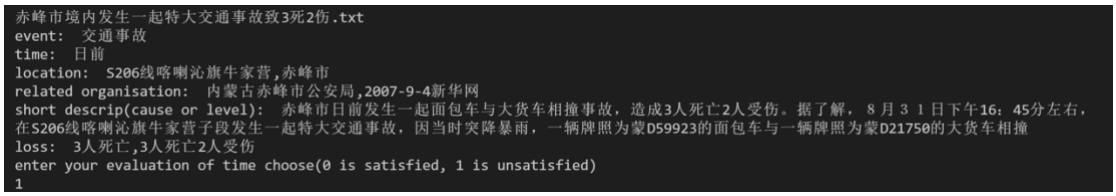
由于一篇文章可能有多个时间，如报道时间、事件发生时间等，而事件的发生只能有一个时间，所以一篇文章的结果输出后对时间的选择人工输入信息抽取对时间的选择的准确性，如果输入 0 则下次也选择出现在这一顺序的时间，如果为 1 则下次选择下一顺序的时间。

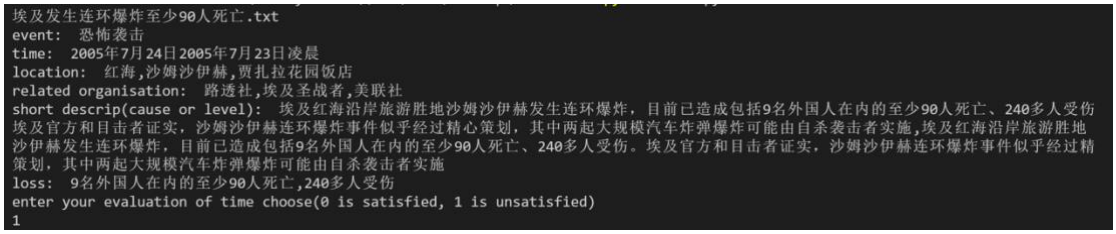
实验结果

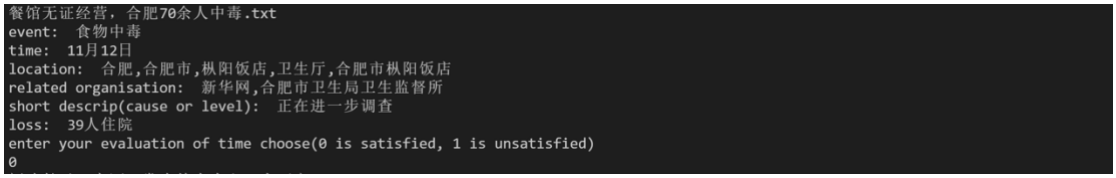
由于数据集过大，实验结果只选择每类一篇文章。











实验结论

事件类型、事件、地点、相关组织、简要描述、伤亡情况信息均抽取成功。对时间的选择，前两次都人工评价为对该时间抽取满意，所以都是文章中的第一个时间，第三次评价为不满意，下一次抽取时间选择文章出现的第二个时间，第四次评价为不满意，下一次抽取时间选择文章出现的第三个时间。实验结果正确。

创新性思考和多媒体信息抽取

本实验中通过自己设计正则表达式匹配文章相关信息和伤亡情况。信息抽取过程中，综合词

性分析和正则表达式模式匹配，从而抽取信息点。如下图：
词性分析：

```
for item in self.nlp_result:
    if item[0] in fire_trigger_list:
        cnt['fire'] += 1
        if item[0] not in fire_set:
            fire_set[item[0]] = 1
        else:
            fire_set[item[0]] += 1
```

正则匹配：

```
pattern = re.compile('[一二三四五六七八九十].[一二三四五六七八九十]级|\\d\\.\\d级')
patterns.append(pattern)
return patterns
type == TRAFF:      BruceHo202, 2 weeks ago • Add files via upload
patterns = []
pattern = re.compile('(?<=[。, ])(?:.*)撞上(?:.*)(?=[。, ])')
patterns.append(pattern)
pattern = re.compile('(?<=[。, ])(?:.*)(?:.*)相[撞碰]')
```

```
patterns.append(re.compile(r'(\d+(?:\w+)死亡)'))
patterns.append(re.compile(r'(\d+(?:\w+)身亡)'))
patterns.append(re.compile(r'(\d+(?:\w+)受伤)'))
patterns.append(re.compile(r'(\d+(?:\w+)烧伤)'))
patterns.append(re.compile(r'(\d+(?:\w+)坠楼身亡)'))
patterns.append(re.compile(r'(\d+(?:\w+)遇难)'))
patterns.append(re.compile(r'(\d+(?:\w+)被灼伤)'))
patterns.append(re.compile(r'(\d+死)'))
patterns.append(re.compile(r'(\d+伤)'))
patterns.append(re.compile(r'[一二三四五六七八九十]+死'))
patterns.append(re.compile(r'[一二三四五六七八九十]+伤'))
patterns.append(re.compile(r'\d+(?:\w*)住院'))
```

在多媒体（如视频、图片等）信息的抽取中，无法使用分词、词性标注进行信息抽取时，可以通过文件特征进行信息抽取。如对图片的信息抽取，可以通过训练提取人物图像、背景等信息，通过文件数据内容提取时间、拍摄设备等。对视频的信息抽取可以对视频所有帧进行图像的信息抽取。