基于知识库的问答实践说明文档

1 简介

本次实践参与者为 WYX 和 YBT，完成了基于给定的三国演义数据集构建本体和知识库，并实现了一个基于该知识库的问答系统。

本次实践中，我们使用 Protégé 工具建立本体，使用 Java 语言和 Apache Jena 工具处理数据、生成三元组。我们将三元组导入到 Virtuoso 数据库中，使用 Python 语言搭建基于模板的问答系统，此外，我们还对可能出现的几种异常情况进行了处理，并使用了 Flask 服务器与 Bootstrap 框架构建了美观的图形界面。

在接下来的几个章节中，第 2 章将介绍构建“三国演义”知识库的思路和过程，第 3 章将介绍实现问答系统的思路和过程，第 4 章将对构建的知识库进行说明和展示，以及对问答系统的运行说明和运行效果展示，第 5 章将给出总结。

2 构建知识库

构建知识库涉及到本体结构构建和三元组生成两部分，下面将分别阐述。

2.1 本体结构构建

根据我们对给定数据文件“person.csv”、“event.csv”以及需要回答的问题的观察，容易首先想到需要构建“人物”、“事件”这两个类；由于问题 5 涉及到势力的效忠者，因此最好也建立“势力”这一实体类别，且因为某些势力为个人而不是政权，因此该类取名为“势力”。由于问答未有涉及，为简明起见，本文未将时间和地点相关信息单独建立实体类别，即均作为数据属性，且其值域均为“rdf:string”。最终，我们构建的本体结构如表 1 所示。

在实践过程中，我们选用 Protégé 工具构建本体，设置的 URL 前缀为“http://ws.nju.edu.cn/tcqa#”，并最终将设计的本体导出为 OWL 文件。

2.2 三元组生成

我们利用 Apache Jena 工具包来实现本体文件的读取、个例和属性的添加以及三元组的导出。具体实现过程如下所述。

1) 读取本体 OWL 文件：建立模型，如图 1 所示。

2) 读取“person.csv”文件：首先需要将原始文件的数据格式规范化，因此对读取的每一行文本都需删除掉多余的符号，比如“,”、“{”等，提取出关键信息，如图 2 所示。然后依据读取的人物姓名来创建一个姓名 URL，再利用本体类的 createIndividual()方法创建这个姓名 URI 的本体实例 Individual。此处需要注意的地方是，原始文件中存在一些重名的人（比如刘氏[曹操妾]和刘氏[曹爽妾]），因此字符串中方括号（如果存在的话）的内容也要包括到姓名值中。

表 1 本体结构

实体名称	属性名称	属性类别	值域
人物	姓名	data property	rdf:string
	字	data property	rdf:string
	性别	data property	rdf:string
	拼音	data property	rdf:string
	介绍	data property	rdf:string
	是否虚构	data property	rdf:string
	出生时间	data property	rdf:string
	死亡时间	data property	rdf:string
	古代籍贯	data property	rdf:string
	现代籍贯	data property	rdf:string
	效忠势力	object property	势力
事件	事件名	data property	rdf:string
	章节	data property	rdf:string
	时间	data property	rdf:string
	发生地点	data property	rdf:string
	简述	data property	rdf:string
	提示	data property	rdf:string
	历史	data property	rdf:string
	人物	object property	人物
势力	势力名	data property	rdf:string

```
// 本体模型工厂类的静态函数创建本体模型model
OntModel model = ModelFactory.createOntologyModel();
// 将本体文件读入到本体模型model中
InputStream ontologyIn = FileManager.get().open(OWL_file);
model.read(ontologyIn, "RDF/XML");
```

图 1　读取本体 OWL 文件

```
// CSV格式文件为逗号分隔符文件，这里根据逗号切分
String item[] = line.split(", ");
// 对每一个item的内容，将属性名和值提取出来
for (int i = 0; i < item.length; i++) {
    item[i] = item[i].replace("{", "")
      .replace("\'", "").replace(":", "").replace("}", "");
}
```

图 2　读取“person.csv”文件删除多余符号

添加 Individual 后，然后依次读取一行中剩下的属性并添加相应属性值。此处也需要注意，因为人物属性中包含效忠势力这一个 object 属性，因此如果某一效忠势力 Individual 还没有创建，需要在这里进行创建。

3) 读取 event.csv 文件

同样的，首先对读取的文本每一行做数据格式处理。然后依据读取的事件名来添加事件类的一个 Individual，之后继续读取和添加剩下的属性和属性值。此处需要注意的是因为事件类包含涉及人物这个对象属性（object property），在计算一个事件包含的涉及人物个数时还需要一个循环来处理。

4) 导出三元组

将最后构造完成的知识库数据使用“model.write()”函数以 RDF/XML 格式写入文件输出流，分别以 OWL、N-Tripples、TURTLE 的格式导出到文件中。

3 构建基于知识库的问答系统

该问答系统采用基于模板的方式识别解析和回答问题。首先使用正则表达式匹配识别出问句中的槽，并确定对应的问题模板，然后生成对应的 SPARQL 语句，查询知识库，随后对查询结果进行处理并显示。值得提及的是，除了基本的模板识别匹配，本文还对可能出现的几种异常情况进行了处理，并制作了美观的图形界面，提升使用体验。关于问答系统，本文将从以下几个方面进行介绍。

3.1 槽识别、模板匹配与 SPARQL 语句生成

该部分主要使用正则表达式匹配作为核心技术，识别槽对应的属性值并匹配对应的问题模板。每一种类型的问题对应一个问题模板，每个模板中明确该问题对应的正则表达式、生成 SPARQL 语句的模板、用于回答该问题的自然语言等。以问题 1（事件"张角受仙人赠书点化"出现在三国演义第几章节中？）对应的模板（如图 3 所示）作为示例，对几个关键的字段进行说明。

```
templates = {
  1: {
    'name': '事件章节',
    'info': '给出指定事件在书中出现的章节',
    'slot_list': ['event_name'],
    'regex': ANY_SLOT_0 + QM + event_name + QM + ANY + CHAP + ANY,
    'slot_score':{'ANY_SLOT_0': {'事件': +200,'人物': -100,'势力': -100}},
    'type': 'select',
    'sparql_expression': '''
        ?a rdf:type :事件.
        ?a :事件名 '{event_name}'.
        ?a :章节 ?x.
    ''',
    'reply': '事件“{event_name}”出现在书中的{value}。',
    'none_reply': '没有找到事件“{event_name}”在书中出现的位置。',
    'checks': ['event_name']
  },
  ……
}
```

图 3　问题模板示例

所有问题模板（templates 变量）为一个字典，每个问题的模板为其中的一项，也是字典类型的数据结构。第 1 个模板中：

a) “slot_list”指定该问题的（实体）槽列表；

b) “regex”指定正则表达式。本文使用 REfO 工具，因此图中第 6 行以该形式给出正则式，它的标准形式为“.*"{event_name}".*章节.*”；

c) “slot_score”指定各个关键字的得分。在正则式中匹配任意内容的 ANY 字段（“.*”），由于某些问题模板比较相似，ANY 字段的内容对于区分不同的模板至关重要，因此该字段规定 ANY 字段中出现不同关键字的得分，以作为确定模板的另一依据。

d) “type”字段指定该问题对应的 SPARQL 语句的查询类型，对于本问答系统有两种，即“select”（列举）和“count”（计数）；

e) “sparql_expression”指定对应 SPARQL 语句中“where”括号中的语句的模板；

f) “reply”指定回答该问题的自然语言模板；

g) “none_reply”和“checks”与异常情况处理有关，它们会在下一节详细说明。

当得到问句后，首先依次使用每个问题模板的“regex”正则式对句子进行匹配，如果该模板能匹配，则对应位置的槽的属性值也同时获取到，将匹配结果加入到候选列表。当匹配完所有模板后，将候选列表中的模板按照匹配得分从大到小排序，选择分数最高的模板作为最终匹配的模板。最后，使用实体属性值格式化 SPARQL 语句模板，生成最终的 SPARQL 查询语句。

3.2 SPARQL 查询与解析

我们将第 2 章生成的三元组导入到 Virtuoso 数据库中，并利用 Virtuoso 提供的接口以及 Python 的 SPARQLWrapper 工具查询知识库，查询结果以 JSON 格式返回，结果是 URI 长字符串。我们进一步对结果进行解析，提取其中用于回答问题的字段，并以模板中“reply”字段的模板进行格式化，得到完整的自然语言答案。

3.3 异常处理

为了使系统的使用体验更好，本文对以下几种可能出现的异常情况进行了处理，对于不同的情况使问答系统给出不同的回复。

3.3.1 无法理解问题

当所有的模板均无法匹配问句时，则认为系统无法理解问句，系统将不再进行后续操作，直接返回错误答复“对不起，我无法理解您的问题，请换种说法问我吧！”

3.3.2 查询无结果

当通过 SPARQL 查询知识库没有得到有效结果，即对于“select”查询其结果为空，或对于“count”查询（select count(•)）其结果为 0 时，系统会进一步查询知识库中是否存在用户所输入的实体属性（事件名称、人物姓名或势力名称），目的是验证它们是否正确，我们称该步骤为实体验证，并根据验证结果给出不同

的错误提示。具体操作逻辑如图 4 所示（仅体现处理逻辑，实现方式请参看源代码）。

```
if result_exists:  # 如果有结果
    # 正常返回查询结果
else:  # 如果没有查询到有效结果
    checks_results = []  # 所有验证结果的错误提示的列表
    checks = template['checks']  # 从问题模板中获取需要验证的实体属性
    for prop in checks:  # 对于每一个需要验证的属性
        check_template = checks_templates[prop]  # 获取验证属性的查询模板
        check_sparql = generate_check_sparql(check_template, arguments)
        # 获取用于验证的SPARQL语句
        check_result = query(check_sparql)  # 查询知识库
        check_result = parse(check_result)  # 解析验证查询的结果
        if check_result is None:  # 验证查询也没有结果
            check_reply = get_check_reply(prop)  # 获取对应的提示
            checks_results.append(check_reply)  # 将提示加入到列表中

    if len(checks_results) != 0:  # 列表中有验证结果提示
        overall_reply = generate_overall_reply(checks_results)
        # 给出属性名称错误的提示
    else: # 验证查询有结果
        none_result_reply = template['none_reply']
        # 给出问题没有结果的相应提示
```

图 4　实体验证操作

当没有查询到有效结果时，对于当前问题模板中“checks”字段指定的所有实体（即需要验证的实体）依次验证。首先根据不同于问题模板的另一套模板“checks_templates”（规定验证实体使用的 SPARQL 语句模板、查询类型、对应回复语句，具体见代码）和之前识别的实体属性值生成验证查询知识库的 SPARQL 语句，例如，现在正在进行“事件”实体的验证，从之前的槽识别中得到用户输入的事件是“桃园三结义”，那么则生成查询“知识库中是否有名称为‘桃园三结义’的事件”的 SPARQL 语句。然后向知识库发送验证查询请求，并解析查询结果。如果结果显示，当前验证的实体在知识库中也没有，则说明很可能是用户将实体的名称输错了，例如问题中所涉及的事件名称输错了，则生成该实体对应的错误提示，并加入到列表。

当程序完成对于所有实体的验证，如果存放错误提示的列表不为空，则说明至少有一个实体的名字被输错了，则生成一个全面的提示，返回给用户，例如“对不起，没有查询到结果。没有查询到名为‘桃园三结义’的事件，没有查询到名为‘曹草’的人物，请检查您的提问是否有误。”

如果列表中为空，则说明对于实体的验证都通过了，也就是说，很可能不是用户输错了实体名称，而是知识库中确实没有相应信息，则根据当前所使用的问题模板“none_reply”对应的字段，给出相应提示，如对于图 1 所示的问题 1 的模板，就会给出例如“没有找到事件‘桃园三结义’在书中出现的位置”的提示。

3.3.3　服务器查询出错

当出现无法连接知识库时，程序的查询过程就会抛出异常。我们对这种情况进行处理，返回“抱歉，我的服务器出现了某些错误，请稍后重试。”的提示。

3.4 各类以及主要函数的说明

问答系统使用 Python 开发，实现的几个类及其主要函数如下：

1) SlotRecognizer：使用问题模板对句子进行匹配

a) recognize(sentence)：以问句作为输入，依次使用每个问题模板对问句进行匹配，返回匹配结果，包括匹配度最高的模板 ID 以及对应的属性。

b) match(sentence, template)：给定问句和问题模板，使用模板对问句进行匹配，返回匹配结果，包括是否能匹配的布尔值、实体槽属性值和匹配得分。

2) SparqlGenerator：根据模板中的 SPARQL 语句模板和匹配得到的属性值生成 SPARQL 查询语句

3) QueryManager：负责处理问句、查询知识库和处理结果等一条龙服务

a) ask(ques)：一条龙服务，以问句进行输入，对问句文本进行简单的处理，并调用 SlotRecognizer 对象进行模板匹配、调用 SparqlGenerator 对象生成 SPARQL 语句，并调用 query 成员函数查询知识库，调用 parse_result 成员函数解析查询结果，并对异常情况进行处理，返回问题的结果答案。

3.5 图形界面

本文使用 Flask 微型服务器和 Bootstrap UI 框架开发了对话系统的图形界面，该框架基于 HTML、CSS 和 JavaScript 技术。用户在聊天窗口发送问题，JavaScript 脚本对文字进行处理，并使用 AJAX 异步传输给 Flask 服务器，服务器调用 QueryManager 类的对象进行问答系统的后端操作，并返回问题的答案。JavaScript 脚本接收答案并显示。由于图形界面不是本文重点，这里就不详细介绍，后文将会有效果的展示。

4 成果说明和展示

4.1 知识库构建

知识库构建的成果和代码位于文件夹“附件 1. 知识库构建”中，其中：

a) “RDF.owl”为 OWL 文件；

b) “RDF_n_triples”为 N-Tripples 文件；

c) “RDF_turtle”为 TURTLE 文件。

d) “generate_triples.java”为处理数据和生成三元组的 Java 代码。

三种格式的三元组数据展示如图 5、图 6、图 7 所示。

4.2 问答系统

问答系统的工程代码位于“附件 2. 问答系统”目录下的“R3K-KBQA”目录。

```owl
<人物 rdf:about="http://ws.nju.edu.cn/tcqa#司马钧">
    <字>叔平</字>
    <现代籍贯>河南焦作市温县西</现代籍贯>
    <死亡时间>? </死亡时间>
    <是否虚构>史实人物</是否虚构>
    <拼音>sī-mǎ-jūn</拼音>
    <古代籍贯>司隶河内郡温</古代籍贯>
    <介绍>那七庙：汉征西将军司马钧，钧生豫章太守司马量，量生颍川太守司马隽，隽生京兆尹司马防，防生宣帝司马懿，懿生景帝司马师、文帝司马昭，是为七庙也。</介绍>
    <性别>男</性别>
    <姓名>司马钧</姓名>
    <出生时间>? </出生时间>
    <效忠势力>
        <势力 rdf:about="http://ws.nju.edu.cn/tcqa#东汉">
            <势力名>东汉</势力名>
        </势力>
    </效忠势力>
</人物>
```

图 5　OWL 数据展示

```n-triples
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#字> "None" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#古代籍贯> "None" .
<http://ws.nju.edu.cn/tcqa#成济> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://ws.nju.edu.cn/tcqa#人物> .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#是否虚构> "史实人物" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#现代籍贯> "None" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#介绍> "贾充心腹,在贾充的指使下杀死曹髦,反被司马昭杀." .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#拼音> "chéng-jì" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#性别> "男" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#姓名> "成济" .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#出生时间> "? " .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#效忠势力> <http://ws.nju.edu.cn/tcqa#魏> .
<http://ws.nju.edu.cn/tcqa#成济> <http://ws.nju.edu.cn/tcqa#死亡时间> "260" .
```

图 6　N-Triples 数据展示

```turtle
:朱异      a          :人物 ;
  :介绍      "诸葛诞叛乱之时，奉丞相孙綝之命担任援军的先锋。接连打败仗，惹得孙綝大怒，败给司马昭后被处斩。" ;
  :出生时间    "? " ;
  :古代籍贯    "扬州吴郡吴" ;
  :姓名      "朱异" ;
  :字        "季文" ;
  :性别      "男" ;
  :拼音      "zhū-yì" ;
  :效忠势力    :吴 ;
  :是否虚构    "史实人物" ;
  :死亡时间    "257" ;
  :现代籍贯    "江苏苏州市" .
```

图 7　TURTLE 数据展示

4.2.1　运行说明

运行问答系统，请按照以下步骤操作：

1) 将三元组导入 Virtuoso 数据库

Graph 的名称设为“R3K”。你也可以命名为其他名称，只需在“config.py”文件中做相应的修改。具体操作略。

2) 安装 Python 所需要的库

本工程所需要的库为包括：REfO、SPARQLWrapper、Flask，可以使用以下命令进行安装：

pip install refo sparqlwrapper flask

3) 运行程序

通过命令“python app.py”运行具有图形界面的 Flask 程序。你也可以通过命令“python cli_app.py”运行基于命令行界面的问答程序。

4.2.2 运行展示

1) 问答功能

界面及问答功能的展示如图 8 所示。

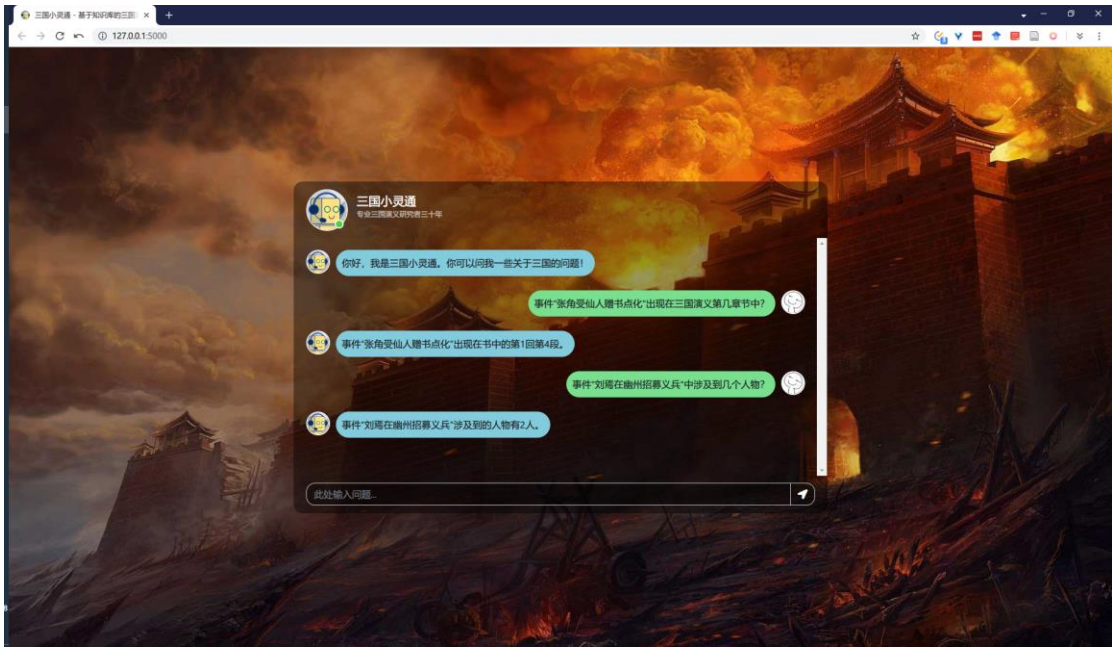

图 8　系统界面和问答功能展示

系统针对 5 个给定问题的回答如下：

a) 事件"张角受仙人赠书点化"出现在三国演义第几章节中？

　　答：事件“张角受仙人赠书点化”出现在书中的第 1 回第 4 段。

b) 事件"刘焉在幽州招募义兵"中涉及到几个人物？

　　答：事件“刘焉在幽州招募义兵”涉及到的人物有 2 人。

c) 事件“刘关张桃园三结义”中涉及人物是哪几个？

　　答：“事件刘关张桃园三结义”涉及到的人物有关羽、刘备、张飞。

d) “崔烈”参与了哪些事件？

　　答：人物“崔烈”参与的事件有：长安城破王允被杀。

e) 效忠“魏”的人物有多少？

　　答：势力“魏”的效忠者有 358 人。

2) 异常处理

系统能够对几种可能出现的异常情况进行处理，具体情况如下所述。

a) 当系统不能理解用户的问句时，将会给出“无法理解”的回复，如图 9 所示。

图 9　无法理解问题时的答复

b) 当用户将实体属性输错导致系统无法查询出有效结果时，系统会进一步查询是否有该实体，并给出相应回复，如图 10 所示。

图 10　当查询不到有效信息时的答复

c) 当服务器出错（比如数据库无法连接时），系统会给出“服务器出错”的回复，如图 11 所示。

图 11　服务器出错时的回复

5　总结

本次实践中，我们边学边做、学以致用，学习了构建本体、构建知识库的一系列操作，对知识库、三元组和知识图谱有了比较清晰地认知，并对基于知识库和模板的问答有了一定的了解。总的来说，收获颇丰。

感谢 HW 老师和助教对课程的付出。