问题生成器接口文档

丁如江 2019. 6. 29

一. 概述

逻辑内核调用问题生成器的大体流程就是逻辑内核喂给生成器逻辑变量，生成器根据逻辑变量生成对应的问题返回给内核，逻辑服务器将问题传送给前段端，前端在将答案发送给逻辑服务器，逻辑服务器调用逻辑内核将答案再输入给生成器，生成器根据答案来确定逻辑变量的取值，然后将逻辑变量以及其取值一同返回给逻辑服务器，具体流程如下：

1. 逻辑内核需要确定一个或多个逻辑变量的取值，会将待确定的逻辑变量放在一个list中（`logic_variable_list`），比如`['咳嗽', '发热']`，然后调用问题生成器的接口将该逻辑变量列表传送给生成器（假设该接口叫做`generateQuestionsbyLogicVaribales`）
2. 生成器的`generateQuestionsbyLogicVaribales`接口解析逻辑变量的列表
   - 列表长度为1，分析逻辑变量的类型，
     - 如果是主述，比如列表为`['咳嗽']`，就可以设计单选题，示例如下：

       $$\text{请问您是否有咳嗽？}(\text{单选})$$
       $$A.\text{有}\ B.\text{没有}$$

     - 如果是查体，比如列表为`['咽喉红肿']`，就可以设计查体题
     - 如果是检查，比如列表为`['血红蛋白含量大于80g/ml']`,就可以设计检查题
   - 列表长度大于1，即为多选题 。比如列表为`['咳嗽','发热']`，就可以设计多选题，示例如下：

     $$\text{请问您是否存在咳嗽和发热的症状？（多选）}$$
     $$A.\text{咳嗽}\ B.\text{不咳嗽}\ C.\text{发热}\ D.\text{不发热}$$

3. `generateQuestionsbyLogicVaribales`返回对应的问题(问题的结构参考《分诊内核接口文档v1.1》,主要就是包装成一个`Question`实例)
4. 逻辑内核将问题列表通过逻辑服务器发送给前端
5. 前端将答案返回给服务器，逻辑服务器在将答案传送给逻辑内核（答案的结构参考《分诊内核接口文档v1.1》，就是一个`Answer`实例))
6. 逻辑内核调用问题生成器的接口将答案传送给生成器（假设该接口叫做`determineValueofLogicVariables`）
7. `determineValueofLogicVariables`解析问题的答案，来确定`logic_variable_list`内逻辑变量的取值，然后返回给逻辑内核

二. 接口定义

基于之前的流程，问题生成器需要提供两个接口

- 根据逻辑变量列表确定问题列表的接口`generateQuestionsbyLogicVaribales`
- 根据问题答案确定逻辑变量取值的接口`determineValueofLogicVariables`

```
class QuestionGenerator:
```

```
    def __init__(self,variable_to_entity_dict):
        '''
        类的构造函数
        :param variable_to_entity_dict:dict类型，是逻辑变量名到kb实体名的映射，逻辑内核调用生成器模块的时候传入
        '''
    def generateQuestionsbyLogicVaribales(self,logic_varilable_list):
        '''
        :param logic_variable_list:list类型，list内是一个个字符串，表示逻辑变量的名字
        '''

        ###分析logic_variable_list来生成对应的问题
        '''
        问题实例，主要就是返回一个包含一个或多个问题实例的列表(当前阶段一般一个list中只有一个Question实例)，Question类的定义附在后面，生成器模块内部定义一下Question类
        '''
        q = Question(
            question_id=1,
            tite='请问您是否有发热',
            question_entity='发热',
            question_type = 1,
            question_options=[{'num':1,'value':'有'},{'num':2,'否'}]
        )
        question_list = []
        question_list.append(q)
        return question_list
    def determineValueofLogicVariables(self,answer_list):
        '''
        :param answer:list类型，list内是一个个Answer的实例（当前阶段一个这个list中一般只有一个Answer实例），Answer类的定义附在后面，生成器模块内部定义一下Answer类
        '''
        ### 分析answer来给逻辑变量赋值

        ###逻辑变量的取值列表，列表内是一个个二元的tuple,tuple[0]表示逻辑变量，tuple[1]表示逻辑变量的取值（1表示True,0表示False,2表示未知）
        variable_value_list = [('发热', 1)]
        return variable_value_list
```

其中会用到的`Question`类，`Answer`类，因为之前是由逻辑内核进行问题的生成，而现在都通过问题生成器来生成问题以及解析答案，所以这两个类应该定义在生成器模块，他们的定义如下：

```
class Question:
    def __init__(self,question_id,title,entity,question_type,
                 options=None,other=False,note="",location=""):
        #问题的id
        self.id = question_id
        #问题的题面
```

```
        self.title = title
        #问题对应的医学实体,也就是对应的逻辑变量名
        self.entity = entity
        #问题的类型
        self.type = question_type
        #问题选项列表
        self.options = options
        #问题是否开启‘其他’选项
        self.other = other
        #问题的补充说明
        self.note = note
        #如果问题是部位选择时，该属性表示被选部分的代码
        self.location = location

class Answer:
    def __init__(self,question_id,options):
        #该答案对应问题的id
        self.question_id = question_id
        #患者选择的选项
        self.options = options
```

而question_type一共有七种，每种的编号如下

问题类型	对应编号
单选	1
多选	2
自主输入	3
文件上传	4
部位选择	5
查体	6
检查化验	7

三. 测试路径

对于七种类型的问题，其实现在逻辑内核只会使用到单选问题，但是为了以后的扩展需要，我们需要把七种问题类型都测试一下，所以我们会构造一些**虚拟数据来**测试这七个问题类型。具体的测试路径如下：

- 首先逻辑内核会发送`['咳嗽']`逻辑变量列表给生成器，生成器会生成一个单选问题
- 之后经由一系列的过程，生成器会将'咳嗽'这个逻辑变量赋好值返回给逻辑内核：
    - 如果是`[('咳嗽', 1)]`,逻辑内核下一步会进行一个查体项目的询问，就是会发送`['咽喉红肿']`给生成器
        - 如果返回`[('咽喉红肿',1)]`,下一步会生成多选题，即逻辑内核会发送`['胸闷', '气短']`给生成器
            - 返回结果是`[('胸闷',1),('气短',1)]`,逻辑内核下一步会让生成器生成自主输入类型的问题，逻辑内核会发送`['自主输入']`给生成器，**这里只是假数据，生成器可以构造一个如下代码的问题返回（只是示例）**，这个问题后结束诊断

```
q = Question(
        question_id=1,
        question_tite='自主输入测试',
        question_entity='自主输入',
        question_type = 3 #3是自主输入的问题类型编号
    )
```

- 返回结果是`[('胸闷',1),('气短',0)]`，逻辑内核下一步会让生成器生成文件上传类型的问题，逻辑内核会发送`['文件上传']`给生成器，**这里只是假数据，生成器可以构造一个如下代码的问题返回（只是示例）**，这个问题后结束诊断

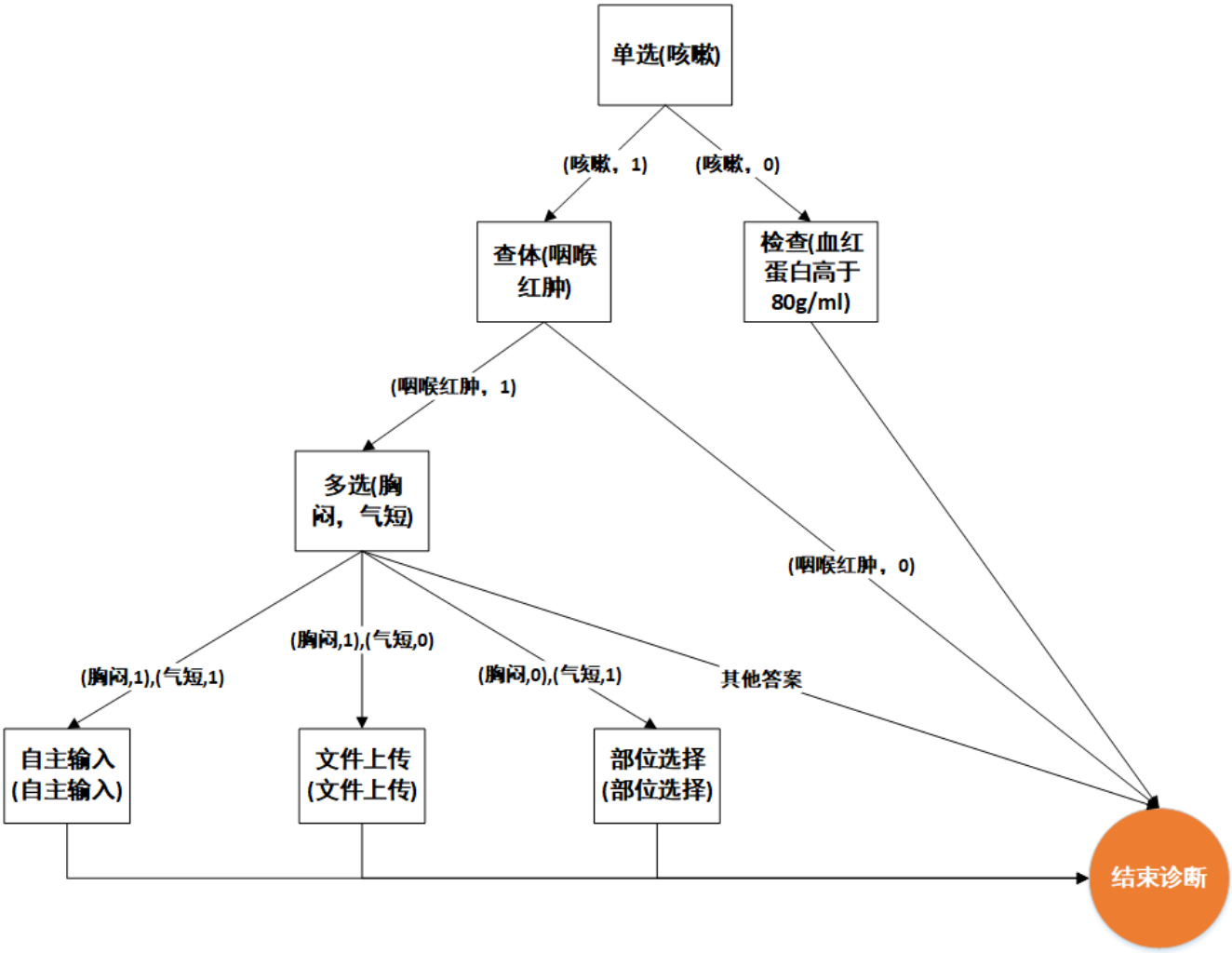
```
q = Question(
        question_id=2,
        question_tite='文件上传测试',
        question_entity='文件上传',
        question_type = 4 #4是文件上传的问题类型编号
    )
```

- 返回结果是`[('胸闷',0),('气短',1)]`，逻辑内核下一步会让生成器生成部位选择类型的问题，逻辑内核会发送`['部位选择']`给生成器，**这里只是假数据，生成器可以构造一个如下代码的问题返回（只是示例）**，这个问题后结束诊断

```
q = Question(
        question_id=3,
        question_tite='部位选择测试',
        question_entity='部位选择',
        question_type = 5 #5是文件上传的问题类型编号
    )
```

    - 返回结果是其他，直接结束诊断，与生成器的交互结束
  - 如果返回`[('咽喉红肿',0)]`,结束诊断
- 如果是`[('咳嗽', 0)]`,逻辑内核下一步会进行一个检查项目的询问，就是会发送`['血红蛋白高于80g/ml']`给生成器
  - 无论其返回结果如何，结束诊断

测试路径图如下：


单选(咳嗽)
(咳嗽，1)
(咳嗽，0)
查体(咽喉红肿)
检查(血红蛋白高于80g/ml)
(咽喉红肿，1)
多选(胸闷，气短)
(咽喉红肿，0)
(胸闷,1),(气短,1)
(胸闷,1),(气短,0)
(胸闷,0),(气短,1)
其他答案
自主输入(自主输入)
文件上传(文件上传)
部位选择(部位选择)
结束诊断