# CoolQRobot

## By NJUMSC

实体回复部分维护手册

Entities Reply

By NJUMSC 高若朋

邮箱：hellorpg2017@gmail.com

№ 所有关于实体（Entities）回复匹配以及回复的代码段几乎都在文件 EntitiesSwitch.py 中，如果没有需要，请在对于此部分的维护中不要更改其他文件的代码，本文件中的代码也请依照下面的指引进行维护。

№ 建议采用 GitHub 或者其他的版本控制和管理系统来维护整个程序的代码。在遇见修改代码之后程序崩溃或者报错等问题，无法解决时，除了咨询开发团队之外，还可以直接回溯版本。

首先需要理解 LUIS 所提供的返回格式。

其返回的是一个数据库的数据条目，我们不方便进行直接的操作，所以需要用到 as_dict( )方法使得其转换为如下的字典返回格式：

```
return 格式如下
{'query': '活动部是干什么的',
'top_scoring_intent': {'intent': '询问活动', 'score': 0.9997649},
'entities': [{'score': 0.9993229,
              'entity': '活动部',
              'type': '名字::部门',
              'start_index': 0, 'end_index': 2}]
}
```

在实体回复部分的代码中，我只涉及了最后一组数据：‘entities’的相关分析。

在 EntitiesSwitch.py 文件的开头，可以看到两段代码：

```
3   def entities_match(data):
4       """函数体，采用switch的方式对传入的data进行分析，识别其中的实体
5          如果其中有和列表中实体相同的实体，则返回索引值
6          否则，返回-1作为异常标志"""
7       entities_list = ["活动部","校运会","技术部","宣传部"]
8       # 可以在entities中增加足够多的实体，来完成不同的实体识别
9       for entity in data:
10          if entity.as_dict()["entity"] in entities_list:
11              return entities_list.index(entity.as_dict()["entity"])
12      return -1
13
14
15  def entities_ans(index):
16      """函数体，采用索引来寻找对应的回答"""
17      ans_list = [
18          "活动部是搞事情的呀！",
19          "校运会的时间是2018.11.2 ~ 2018.11.4",
20          "技术部的都很帅",
21          "宣传部的都是漂亮的小姐姐和小哥哥",
22      ]
23      return ans_list[index]
```

[3~23 行]

这两段代码的作用是在函数说明文档中详细列出的，但是这两段代码是已经在迭代开发的过程中被替代的部分，在他们之下有两段代码会更好的完成相应的功能。

什么？你想打我？对不起，代码是我写的，我就是喜欢留着，你打我呀。

最重要的是下面的两个函数体：

首先是第一个：

```python
def entities_module_match(data):
    """前面的函数虽然实现了一定的功能，但是在同一大类的实体识别的时候，运维十分的繁琐
       或许可以增加一个以大类进行识别的方式"""
    technology_team = ["技术部","技术队","技术家园"]
    activity_team = ["活动部","活动队"]

    entities_module_list = [technology_team, activity_team]

    """
    曾用代码：
    for entity in data:
        if entity.as_dict()["entity"] in entities_list:
            return entities_list.index(entity.as_dict()["entity"])
    """
    # entity_module_find = 0
    power = 0.0000000
    # ret_list = ["NULL"]
    ret_index = -1
    for entity_module in entities_module_list:
        for entity in data:
            if entity.as_dict()["entity"] in entity_module:
                # print(float(entity.as_dict()["score"]))
                # return entities_module_list.index(entity_module) #如果是直接返回的话，忽略了权重这一个指标
                if float(entity.as_dict()["score"]) > power:
                    ret_index = entities_module_list.index(entity_module)
    return ret_index
```

这段函数中，运维所需要添加的实体以 List 的形式存储就可以了。

如上的 technology_team 和 activity_team 就是两个实体的 List，如果需要添加实体的话，只需要使用同样的命名方式，如命名一个 A=[……]，其中的……部分就是该段定义可以响应的实体部分，然后需要将这一部分 A 添加到另一个 List entities_module_list 中，具体表现为：

entities_module_list = [technology_team, activity_team, A]

其中蓝色部分为添加部分，要注意逗号的添加。

№ 注意加入的部分请放在先有的部分的后面（尾插入），以免造成后续的顺序问题。

第二段代码如下：

```
53  def entities_module_ans(index):
54      """与上面的模式化匹配实体相对应，模式化的回应"""
55      technology_team_ans = [
56          "技术部都是帅哥靓女",
57          "技术部的都是大佬",
58          "怎么，据我所知，他们可不会修电脑"
59      ]
60
61      activity_team_ans = [
62          "活动部是搞事情的呀！",
63          "喜欢活动部的小可爱吗？",
64          "悄悄告诉你个秘密，我也是活动部的呢",
65          "我也喜欢活动部呀",
66      ]
67
68      ans_list = [technology_team_ans, activity_team_ans]
69
70      return ans_list[index][random.randint(0, len(ans_list[index]) - 1)]
71
```

这一段是返回了回复用户的语句。

如上可见，两个回复分别对应了技术部和活动部两个实体。

如果需要添加相应的回复，在上述的添加了 List A 之后，我们需要对 A 中的实体进行回复，所以在第二个函数中要创建一个 List 与之对应，建议的命名为 A_ans，如上所示。

添加方法仍是以 List 形式添加。之后将 A_ans 添加到 ans_list 中，同样采用尾插入。

№ 注意，加入的 A_ans 在 ans_list 中的索引要与 A 在 entities_module_list 中的索引是一致的，这样子才能得到相应的答复。

P.S.

关于代码中的疑问或者发现了隐藏的代码 BUG，欢迎按照联系方式进行联系。

由于本人的第一语言是 C++，所以在 Python 编程中可能使用了较为繁琐的方式和逻辑思维，还望海涵。

如果有更好的实现方式，欢迎与开发者团队进行联系和探讨。

感谢配合。