基于ES的知识问答Demo介绍

实践展示

- 本实验：尝试在Elasticsearch实现了一个简单的语义数据检索引擎的基础上，完成一个知识问答Demo。通过解析输入的自然语言查询语句生成Elasticsearch查询，然后执行得到结果。目前实现可以按照实体检索属性，按照多种属性条件的布尔组合检索实体等。实验使用的数据集为由三元组组成的人物属性数据集
- Demo: http://120.27.213.250:9000/

实践展示

http://openkg.cn/tool/elasticsearch-kbqa

实践展示 - 功能

实体查询

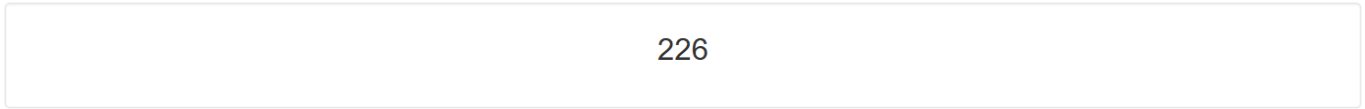
- 查询一个实体，返回该实体的知识卡片。例如“姚明是谁?”，“谁是佟大为？”

属性名称	属性值
birthPlace	上海
gender	男
children	姚沁蕾
职业	运动员 篮球运动员 其他 上海大鲨鱼队老板
birthDate	1980年9月12日
height	226
nationality	中国
subj	姚明
alumniOf	上海体育技术教育学院
民族	汉族

实践展示 - 功能

实体的属性查询

- 询问一个实体的某个属性，返回该属性对应的属性值。例如“姚明有多高？”，“姚明是干什么的”



多跳查询

- 询问从一个实体出发的多个属性，在知识库里多跳查询。例如“姚明的女儿的母亲是谁？”，“姚明的女儿的母亲的身高是多少”

实践展示 - 功能

按照多种属性条件检索实体

- 查询满足多个属性值条件的实体。例如，“中国女运动员”，“身高大于170的中国或美国的作家”，“身高>200,体重小于200的中国篮球运动员”
- 属性值条件可以是等于或者范围查询；多属性条件之间可以是且(AND)和或(OR)关系

实体名称	查询链接
林晴萱	林晴萱
谢莉·杜瓦尔	谢莉·杜瓦尔
卡里姆·阿卜杜拉·贾巴尔	卡里姆·阿卜杜拉·贾巴尔
余秋雨	余秋雨
何炅	何炅
曹德权	曹德权
迈克尔·克莱顿	迈克尔·克莱顿

实现步骤 - 确定索引方式

- 原始数据集：人物属性数据，由三元组组成
  - 属性种类十几种，除height, weight外都是字符串类型

```
A.J.万德 affiliation 篮球
A.J.万德 description A.J.万德（A.J. Wynder），1964年出生，前美国篮球运动员。
A.J.万德 nationality 美国
A.J.万德 weight 82公斤
A.J.库克 birthDate 1978年7月22日
A.J.库克 birthPlace #加拿大安大略省奥沙瓦
```

- 依据多种属性条件检索实体 → 一个实体的所有属性及属性值为一个文档，而不是一个三元组一个文档 → 便于检索，且效率更高
- height, weight等属性须支持范围检索，存储类型为integer，单独作为字段存储
- 其它属性，都存储为keyword类型，且存储为nested object，即不作为单独的字段存储(种类较多)

实现步骤 - 数据格式转换

按照确定的索引方式转化为json文档

```
A.J.万德 affiliation 篮球
A.J.万德 description A.J.万德 (A.J. Wynder), 1964年出生，前美国篮球运动员。
A.J.万德 nationality 美国
A.J.万德 weight 82公斤
A.J.库克 birthDate 1978年7月22日
A.J.库克 birthPlace #加拿大安大略省奥沙瓦
```

```
{
    "subj":"A.J.万德",
    "weight":"82公斤",
    "height":None,
    "po":[{"pred":"affiliation","obj":"篮球"},
          {"pred":"description","obj":"A.J.万德 (A.J. Wynder) , 1964年出生，前美
国篮球运动员。"},
          {"pred":"nationality","obj":"美国"},
         ]
}
```

需要对数据做一些预处理

- 清理属性值中无关字符
- Height, weight等属性统一单位
- “职业”等属性属性值存在多个值的情况，切分成多个属性值对

实现步骤 - 导入Elasticsearch

- 为Elasticsearch新建index和type (mapping文件)
  - Mapping文件指定了文档中每个字段在ES中存储的数据类型和位置

```
#新建第一个index 和 type
curl -XPUT 'localhost:9200/demo?pretty' -H 'Content-Type: application/json' -d'
{
    "mappings": {
        "person": {
            "properties": {
                "subj": {"type": "keyword"},
                "height": {"type": "integer"},
                "weight": {"type": "integer"},
                "po":{
                    "type": "nested",
                    "properties":{
                        "pred":{"type":"keyword"},
                        "obj":{"type":"keyword"}
                    }
                }
            }
        }
    }
}
'
```

- 使用Elasticsearch的insert API将数据导入建立的index和type

实现步骤-属性同义词扩展

- 因为实验的数据集较小，包含的属性种类不多，因此可以人工增加一些同义的属性词
  - 每一行的第一个词是数据集中的属性，后面是添加的同义属性
  - 查询时如果碰到同义属性，映射到数据集中的属性

```
weight 重量 多重 体重
relatedTo 相关 有关
telephone 电话 号码 电话号 电话号码 手机 手机号 手机号码
birthDate 出生日期 出生时间 生日 时候出生 年出生
height 高度 海拔 多高 身高
sibling 兄弟 哥哥 姐姐 弟弟 妹妹 姐妹
workLocation 工作地点 在哪工作 在哪上班 上班地点
children 子女 孩子 女儿 儿子
年龄 几岁 多大
代表作品 代表作 著作 成就 作品
homeLocation 家庭住址 住哪 住在哪 住在什么
```

实现步骤 - 查询解析及构造

解析自然语言查询 → 生成logical form → 生成ES查询 → 执行

- 预定义4种查询类型，解析时将查询分类到其中一种
- 为这4种查询预定义logical form的模板和ES查询的模板，填充模板得到最终ES查询语句

查询类型	自然语言查询语句	Logical form
实体检索	姚明是谁	姚明
实体的属性检索	姚明有多高	姚明:height
实体属性的多跳检索	姚明的女儿的母亲是谁	姚明:女儿:母亲
多种属性条件检索实体	身高大于180的中国或美国的作家	身高>180 And 国籍:中国 Or 国籍:美国 And 职业:作家

实现步骤 – logical form模板

- Logical form 里有如下几个元素：
  - 三元组的成分: S(subject), P(predicate), O(object)
  - 单个属性条件的OP(operator): :, <, >, <=, >=. 例如“职业:演员”,”身高>200”
  - 属性条件之间的与、或关系:, And, Or. 例如“职业:演员 And 身高>200”

查询类型	Logical form模板	示例
实体检索	S	姚明
实体的属性检索	S:P	姚明:height
实体属性的多跳检索	S:P1:P2 …	姚明:女儿:母亲
多种属性条件检索实体	P1 OP O1 And/Or P2 OP O2 …	身高>180 And 国籍:中国 Or 国籍:美国 And 职业:作家

实现步骤 - 自然语言查询解析

- 首先，对于输入的自然语言问句，识别出其中出现在知识库中的实体名，属性名以及属性值
  - 1）分词
  - 2）对于属性，由于种类较少，直接用字典记录知识库中的所有属性，用匹配的方法找出所有属性名
  - 3）对于实体名，将分词的结果(为保证正确性，可以将分词算法的词典替换成所有实体名，分词工具例如jieba分词支持自定义词典)查询elasticsearch，判断是否存在以该词语为实体名的文档，若存在表明该词语是一个实体名
  - 4）对于属性值，由于变化较大，可以采用模糊匹配的方法，也可以采用分词后n-gram检索es的办法，demo中采取后一种。在判断一个短语是属性值后，还要统计该属性值对应的属性名，当查询语句中缺省了属性名，例如“(国籍是)中国(的)运动员”，缺省了“国籍”，就用该属性值对应的最频繁的属性名作为补全的属性名。

实现步骤 - 自然语言查询解析

- 在识别出查询中所有的实体名，属性名和属性值后，依据它们的数目及位置，确定查询的类型，以便映射到的对应的logical form
  - 如果有实体名
    - 如果有多个属性名，那么是属性值的多跳查询
    - 如果有一个属性名，那么判断实体名和属性名的位置及中间的连接词(“是”“在”“的”等)。若实体名在前，则是实体的属性查询，例如“姚明的身高”；若属性名在前，则是依据属性查询实体，例如“女儿是姚沁蕾”
  - 如果没有实体名，则认为是依据属性查询实体
    - 根据所有属性名和属性值位置的相对关系，确定它们之间的对应关系
    - 对于缺省属性名的属性值，补全属性名
    - 对于缺乏属性值的属性名，例如“身高>200”，如果是特殊类型的属性名,例如“身高”，“体重”，则通过正则表达式识别出范围查询的数值；否则，则忽视。

实现步骤 – 生成logical form

- 在对问题分类后，根据对应类型问题的logical form的模板，将识别出的实体名，属性名和属性值填充进去，生成logical form

- 多个属性条件之间存在“And”“Or”等连接条件
  - 如果在查询语句中出现“或”，“并且”等词，则在填充模板时将连接条件更改为对应的“Or”,“And”,缺省为“And”
  - 如果有两个属性值对应同一种属性名，例如“中国以及美国的作家”,那么也将连接条件改为“Or”

实现步骤 – logical form模板匹配流程伪代码

```
def translate_NL2LF(nl_query):
    '''
    args:    nl_query: 自然语言查询语句
    return:    lf_query: logical form查询语句
    '''
    entity_list = _entity_linking(nl_query)                #识别实体名
    attr_list = _map_predicate(nl_query)                   #识别属性名
    if entity_list:                                        #识别到实体
        if not attr_list:                                  #### 问题类别:    查询单个实体 ####
            lf_query = entity_list[0]                      #生成对应的Logical form
        else:
            if len(attr_list) == 1:                        #### 问题类别:    单个实体属性查询 ####
                lf_query = "{}:{}".format(entity_list[0], attr_list[0]) #SP 生成对应的Logical form
            else:                                          #### 问题类别:    多跳查询 ####
                lf_query = entity_list[0]
                for pred in attr_list:
                    lf_query += ":" + pred                 #生成对应的Logical form
    else:                                                  #### 问题类别:    多个属性条件检索实体 ####
        val_d = _val_linking(nl_query)                     #识别属性值
        retain_attr = _map_attr_val(attr_list, val_d)      #根据相对位置找到属性名和属性值的对应关系，剩下没有对应的属性名
        for a in retain_attr:
            if a == 'height' or a == 'weight':
                value = get_value(a, nl_query)             #正则表达式找出数值
                val_d.append((value, a))
        for v in val_d:
            if v in prev or find_or:                       #同类型属性出现过，或者解析到"或者"等词语
                lf_query += ' OR ' + '{}:{}'.format(pred, v)
            else:
                lf_query += ' AND ' + '{}:{}'.format(pred, v)
    return lf_query
```

实现步骤 - ES查询语句模板

解析自然语言查询 → 生成logical form → 生成ES查询 → 执行

- 对于实体属性查询，包括多跳查询，都是先检索实体，然后获取对应的属性
- 对于多个属性条件检索实体，先为每种单个的属性条件创建ES查询的模板，最后组合成完整的查询

细分的查询类型	Logical form	ES模板_部分查询
检索实体	S1	"query":{"bool":{ "must":{ "term":{"subj“ : S1} }}}
属性条件 op为等于	P1 : O1	"query":{"bool":{"must":[{"term":{"po.obj":O1}}, {"term":{"po.pred":P1}}]}}

属性条件 op为范围检索	P1 >= O1	"range":{
		P1:{"gte":O1}
		}

实现步骤 - ES查询语句模板

- 生成每个属性条件对应的部分ES查询语句后，用如下模板合并成完整查询
  - 下表中part_query表示单个属性条件对应的部分ES查询语句

连接条件	Logical form	ES模板_整体
And	P1 OP O1 And P2 OP O2	"query": {"bool":{"must":[part_query1, part_query2]}}
Or	P1 OP O1 Or P2 OP O2	"query": {"bool":{"should":[part_query1, part_query2]}}

实现步骤－ES查询构造

1.实体查询 及 属性查询

- 解析查询中的实体名和属性名，以实体名为keyword检索实体，并解析出答案中属性名对应的属性值。
- 例如查询“姚明”，构造类似如下查询来检索实体
- 查询“姚明：身高”，先检索实体“姚明”，再获取结果中的“身高”属性的值

```
#按实体名字查询
curl -XGET 'localhost:9200/demo/person/_search?&pretty' -H 'Content-Type:application/json' -d'
{
    "query":{
        "bool":{
            "must":{
                "term":{"subj":"姚明"}
            }
        }
    }
}
'
```

实现步骤 - ES查询构造

2. 多跳查询

- 根据实体和第一个属性查询对应的属性值
- 将此属性值作为下一步的实体，根据第二个属性接着查询属性值
- 如此循环，直至结束

实现：循环调用实体属性查询

```
def search_multihop_sp(entity, preds):
    '''
    args :
        entity:  entity name
        preds: list of predicate
    '''
    for p in preds:
        if not entity_in_KB(entity):
            return False
        card = search_single_entity(entity) #调用实体查询
        if not p in card:
            return False
        o = card[p]
        entity = o

    return o
```

实现步骤 - ES查询构造

3. 依据多种属性条件检索实体

- 解析出每个属性条件及其中的操作，即(pred, op, obj)，生成这一部分对应的部分查询
- 例如：“身高 >= 200”对应的部分查询为：

```
{
    "range":{
        "height":{
            "gte":200
        }
    }
},
```

- 检索除height, weight除外的其它存储在nested object中的属性，例如：“国籍：中国”：

```
"nested":{
    "path":"po",
    "query":{
        "bool":{
            "must":[
                {"term":{"po.obj":"中国"}},
                {"term":{"po.pred":"nationality"}}
            ]
```

指定查询nested object

实现步骤 - ES查询构造

3. 依据多种属性条件检索实体

- 依据属性条件间的And，Or关系合并每个属性条件对应的部分查询
- 例如，“职业：篮球运动员 And 身高>=200 And 国籍:中国”:

```
curl -XGET 'localhost:9200/demo/person/_search?&pretty' -H 'Content-Type:application/json' -d'
{
    "query": {
        "bool":{
            "must":[                    must表示And关系
                {
                    "range":{
                        "height":{
                            "gte":200
                        }}},
                {
                    "nested":{
                        "path":"po",
                        "query":{
                            "bool":{
                                "must":[
                                    {"term":{"po.obj":"中国"}},
                                    {"term":{"po.pred":"nationality"}}
                                ]}}}},
                {
                    "nested":{
                        "path":"po",
                        "query":{
                            "bool":{
                                "must": [
                                    {"term":{"po.obj":"篮球运动员"}},
                                    {"term":{"po.pred":"职业"}}
                                ]}}}}
            ]
```

实现步骤 - 执行ES查询

- 解析自然语言查询 → 生成logical form → 生成ES查询 → 执行
- 基于每种查询问题的模板，填充解析时识别出来的实体名和属性名，生成Elasticsearch查询

```
def _search_single_subj(entity_name):      #查询单个实体
    query = json.dumps({"query": { "bool":{"filter":{"term" :{"subj" : entity_name}}}}}) #组装query
    response = requests.get("http://localhost:9200/demo/person/_search", data = query) #查询
    res = json.loads(response.content)
```

- 构造出用户查询对应的Elasticsearch查询后，执行该查询，并解析查询结果

实践 - 知识问答进阶

1. 歧义

- 前述在识别自然语言问句中的实体名，属性名和属性值的时候，没有考虑到歧义问题，即一个自然语言问句可以对应多种解析方式
- 例如，“毕业于浙江大学的学者”，“浙江大学”应当是一个属性值，但是在识别时却被识别成属性值“浙江”和属性名“大学”。
- 解决方案：
- 1) 依据上下文排除错误的解析。在上例中，很明显后一种解析方式会导致多余一个属性名“大学”对应不到属性值，因此可以排除
- 2）依据句法分析或语义解析，例如dependency parsing 或者 semantic role labeling, 依据短语之间的依赖和修饰关系确定属性名和属性值的对应关系
- 3) 生成多个候选结果，让用户选择

实践 - 知识问答进阶

2. 问题匹配到模板

- 前述在问题分类时简单地通过实体名，属性名和属性值之间的位置关系来实现，规则过于简单
- 解决方案：
  - 1）句法分析，将位置关系换成依赖关系
  - 2）可能可以对应到多个模板，可以都执行，或者排序选择一个

3. 查询refinement

- 用户的某些查询语句可能没有意义，或者超出预设的模板之外，或者意义不清晰，可对应到多种logical form。例如“姚明的爱好”，知识库中无“爱好”这一属性；或者“姚明的代表作品”，虽然“代表作品”存在于知识库中，但不是“姚明”的属性，整个查询没有意义
- 解决方案：生成能正确执行且改动最少的查询，或者生成多个候选查询让用户选择。

实践 - 知识问答进阶

4. 查询功能扩展

- 上述只是基于Elasticsearch实现了基本的查询功能，可通过一些扩展使其支持更复杂的查询

1). 别名检索

- 用户在检索实体时，可能输入的不是知识库中存储的实体名，而是其别名或者简称等，例如”周杰伦”→”周董”
- 解决方案
  - 知识库中每个实体在存储时，存储一个alias属性 (如果有)
  - 判断查询类型为检索实体时，除了根据查询语句检索实体名，还要检索包含属性为alias，属性值为查询语句的实体
  - 推而广之，在所有需要检索实体的地方，除了检索实体名，同时检索alias为查询语句的实体
  - 前述实验中依据实体名唯一表示实体，此处可看出不恰当，应以id为唯一标识，而将实体名作为name属性的属性值，同alias同为属性更符合含义

实践 - 知识问答进阶

4. 查询功能扩展

- 上述只是基于Elasticsearch实现了基本的查询功能，可通过一些扩展使其支持更复杂的查询，同时也可以首先完成基于ES的语义搜索进阶，再使用同样的自然语言问句语义解析或基于模板的方法进行匹配和转换。

实践 - 语义搜索进阶

1. 别名检索

- 用户在检索实体时，可能输入的不是知识库中存储的实体名，而是其别名或者简称等，例如”周杰伦”→”周董”
- 解决方案
  - 知识库中每个实体在存储时，存储一个alias属性 (如果有)
  - 判断查询类型为检索实体时，除了根据查询语句检索实体名，还要检索包含属性为alias，属性值为查询语句的实体
  - 推而广之，在所有需要检索实体的地方，除了检索实体名，同时检索alias为查询语句的实体
  - 前述实验中依据实体名唯一表示实体，此处可看出不恰当，应以id为唯一标识，而将实体名作为name属性的属性值，同alias同为属性更符合含义

实践 - 语义搜索进阶

2. 概念检索

- 用户希望检索某一类型的实体，例如所有类型是“篮球运动员”的实体
- 解决方案：每个实体存储一个type属性。用户做概念检索时，将其当作属性为type的属性检索即可

- 推广，考虑类型的包含关系。例如，“运动员”包含“足球运动员”和“篮球运动员”。用户检索“运动员”时，将其拆分成“足球运动员”和“篮球运动员”单独检索再OR合并。
- 解决方案：为知识库中的type关系建立上下位关系。在解析查询时对上位属性进行查询扩展

实践 - 语义搜索进阶

3. 属性值模糊检索或子串检索

- 在前述的实现中，属性值都是作为keyword类型存储，即不分词，只精确匹配。用户在检索时须指定具体属性和属性值
- 用户可能并不知道知识库中存在哪些属性。因此可将字符串类型属性值存储为String类型（分词处理），此时用户可以只输入任意属性值，查询时将用户输入与知识库中的所有属性值做模糊匹配，按照相似度返回实体列表

4. 推广对不同类型属性值的操作

- 在前述实现中，所有属性，除了height和weight存储为integer类型单独存储，支持range search外，其它都看作Keyword类型存储在nested object中，只支持精确匹配
- 可将所有属性按照不同类型分别存在不同的nested object中，例如integer, Date, string, entity. 每种类型的属性支持不同的操作，例如integer, Date支持range search, entity支持多跳查询或逆关系查找

实践 - 语义搜索进阶

5. 反向检索

- 知识库中可能只存在“姚明，女儿，姚沁蕾”,但用户查找“姚沁蕾:父亲”
- 解决方案：每个实体在存储时，对于其存在反向属性的属性，除了将其作为subject,存储其所有的(predicate, object)对，同时也将其作为object，存储其所有的(reverse-predicate, object)对，此时原来的object作为反向关系的subject
  - 姚明:PO[(女儿，姚沁蕾)...]

    RSP[(父亲，姚沁蕾)...]
- 用户在查询SP时，例如“姚沁蕾：父亲”,除了将其作为原本的SP查询外，同时将其作为反向的PO查询，即检索RSP属性中拥有属性值对(父亲：姚沁蕾)的实体
- 同理，用户在查询PO时，同时也做反向的SP查询
- 可见前述在存储时将实体名存储为“subj”不恰当，应存储为“entity”；同时，不仅要存储PO属性对，还要存储SP属性对

实践 - 语义搜索进阶

6. 属性path查询

- 除了支持依据实体查属性的多跳查询，也可以支持依据属性查实体的多跳查询，例如“女儿:星座:白羊座”的实体
- 解决方案：逆向搜索，先检索有最后一个属性和属性值的PO对的实体，即检索所有(星座，白羊座)实体
- 检索所有拥有属性”女儿“，且属性值在(星座，白羊座）实体中的实体

7. 在查询中嵌套path查询

- 除了多个PO属性条件检索实体，也可以在其中嵌套path查询。例如“国籍：中国 and 女儿：星座：白羊座”
- 解决方案：对于每个属性条件，可以是简单的PO，也可以PO path，也可以是反向SP

实践 - 语义搜索进阶

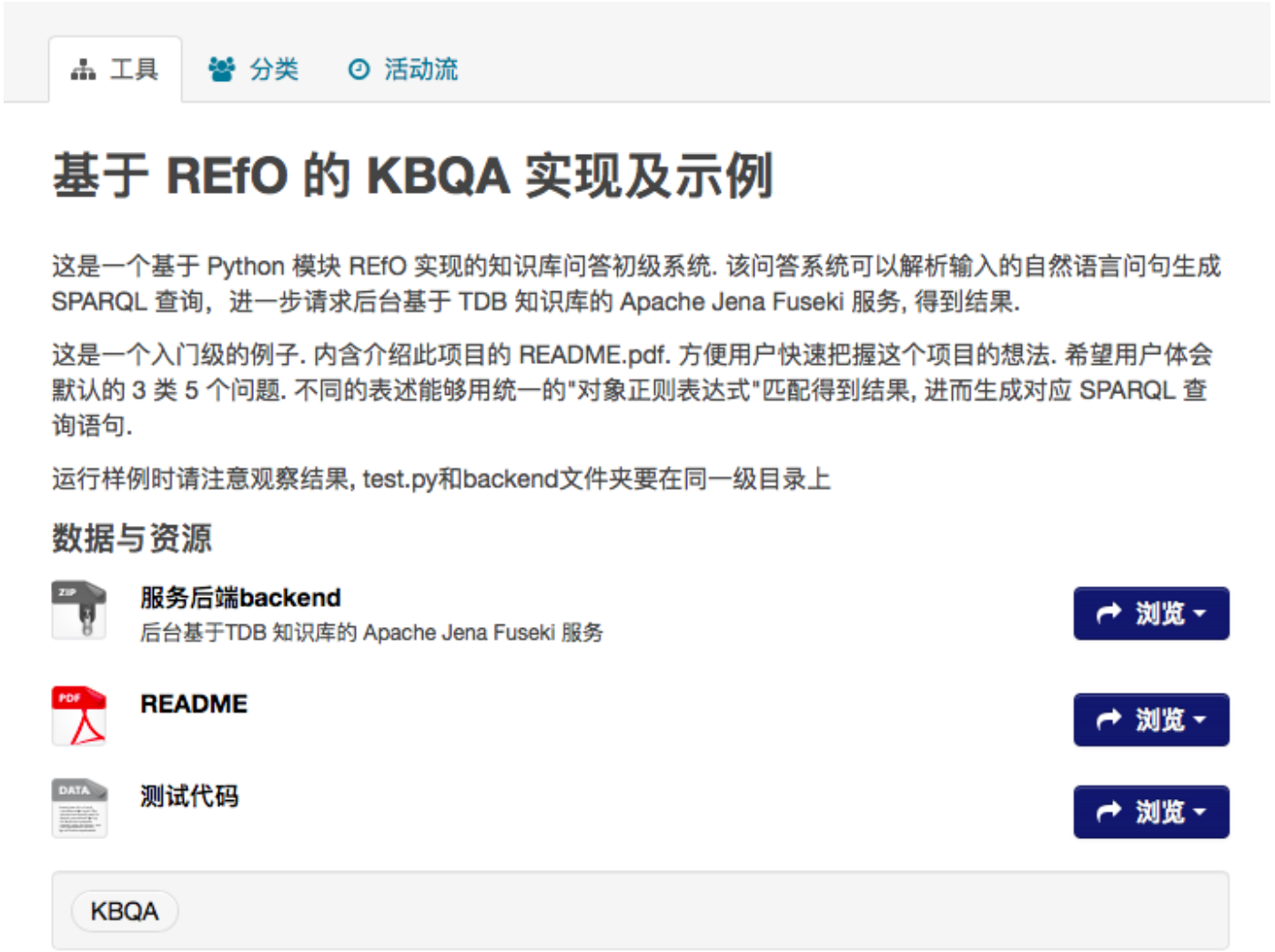
8. 多元关系

- 除了二元关系，也可能是多元关系，例如，“2000年后在火箭队的篮球运动员”，这里不再是“职业：篮球运动员”查实体的二元关系
- 解决方案：对用户查询中的多元关系，同样是将其分解为多个二元关系(类Freebase CVT)的组合查询，上述即“职业：篮球运动员” And “player.member of: 火箭队”And “player.join_time: 2000”

9. 单位统一

- 知识库中的“身高”单位为cm，用户查询“身高>1.9m”
- 解决方案：对用户查询进行预处理，可以用正则表达式等方法识别出带单位的数值，日期等，转化为存储时的单位

实践展示



http://openkg.cn/tool/refo-kbqa

谢谢大家！

本课程课件由OpenKG提供支持

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop