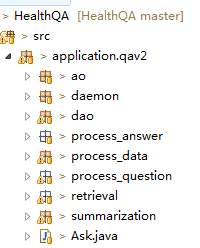
QA安装开发文档

# 代码结构（按照目录）

## 后端目录

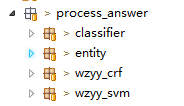
1. 首先看一下application.qav2下的目录结构：



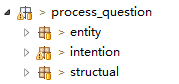
1，ao包下写的是一个个类的属性和方法。以Answer.java为例，类Answer的属性包括id（数据库中唯一ID）、question\_id（回复的问题id）、raw\_content（正文）、piece\_number（片段数）、entity\_content（标注好实体的正文）、seg\_content（分词之后的正文）、question（问题）、kws（关键词）、entities（答案中的实体，也是我添加的用来对答案进行排序）。

2，dao包下是与数据库有关的操作。比如AnswerDAO对应于answer表，HealthqaDB是一些DB操作，获取问题列表，获取答案列表，获取片段列表等。

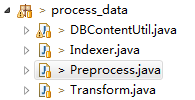
3，process\_answer包，答案处理。**classifier**是片段分类，使用的方法是SVM模型。entity是利用CRF进行实体抽取。wzyy\_crf是最小完整语义切分，因为是对段落进行重新标注标点，所以也是一个序列标注的问题，实现的方法也是CRF。



4，process\_question，问题处理。entity即实体抽取。structual对问题进行结构化表示，使用的方法是正则表达式。Intention问题意图分析，通过疑问词或者疑问句式将问题分为封闭性问题和开放性问题。

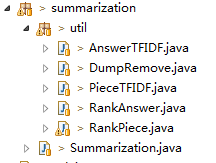


5，process\_data，数据处理。Preprocess.java是从ishc\_data中post表和reply\_post表中获取数据，然后有Transform.java进行处理，形成healthqa中的answer表，question表，piece表，还有entity\_map，它保存在了sqlite数据库中。Indexer是为answer，question，piece用lucene建立索引。

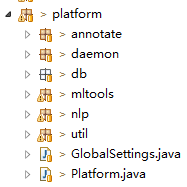


6，retrieval，文本检索，问题检索和答案检索。

7，summarization，摘要。其中，Summarization.java写了findTopAnswers()和findTopPieces()方法，而util则是一些工具包，AnswerTFIDF用来计算答案的tfidf值，PieceTFIDF用来计算片段的tfidf值，RankAnswer和RankPiece记录了打分指标。



1. 我们来看一下platform下的目录结构。



1，annotate标注包，包括按词典标注和按词性标注。

2，Daemon进程守护基类。

3，db，有关数据库的操作，其中dao包本系统不用，database是连接数据库的操作。

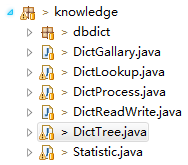
4，mltools是一些机器学习工具包。

5，nlp自然语言处理工具（NLPIR）。

6，GlobalSettings.java 全局环境信息，包括获取工程目录，获取操作系统类型等。

7，Platform.java 主要功能进行分词，segment()。

1. knowledge包下主要是一些辅助的知识库。



本系统用到的有：

1，DictGallary字典仓库，包括药物字典，检查字典，食物字典，治疗字典等等。

2，DictReadWrite，字典的读入读出。

3，DictTree，构建字典树，按字典标注时使用，每一种字典都有一颗树。

## Web目录



app.js：程序启动文件。

bin: 真实的执行程序。

node\_modules:存放所有的项目依赖库。

public：静态文件，包括js脚本，css样式表、image图

routes:路由文件，指定url由某个函数响应。

views：页面文件（jade模板）

package.json:项目依赖配置以及开发者信息。

1，在进行界面优化的过程中，因为设置了答案的三种不同的展示方式，所以我添加了一些内容：

(1)，app.js中，定义路由规则。即添加如下代码





当用/onlyAnswers访问时，调用routes目录下onlyAnswers.js文件；当用当用/onlyPieces访问时，调用routes目录下onlyPieces.js文件；当用/combinePieces访问时，调用routes目录下combinePieces.js文件。

(2)，在routes下添加onlyAnswers.js，onlyPieces.js，combinePieces.js，并指定当有router.get请求时，响应function进行处理，并将结果渲染到相应的jade模板。

(3)，在views下添加onlyAnswers.jade，onlyPieces.jade，combinePieces.jade，对结果显示进行分别处理。

1. 样式的更改

public > stylesheets >style.css中记录了一些样式定义，如果有需要可以在这里进行更改。

1. public > javascripts > my.js 自己写的一些javascript脚本。

# 2 . 数据库设计

1. **question表**

|  |  |  |
| --- | --- | --- |
| 序号 | 字段 | 字段描述 |
| 1 | Id | 对应的post表中问题的id |
| 2 | anno\_content | 标注后的问题内容 |
| 3 | title | 问题的标题 |
| 4 | content | 问题的内容 |
| 5 | structure | 问题的结构化表示文本（JSON格式） |
| 6 | interrogative | 疑问句 |
| 7 | interrogative\_word | 疑问词 |
| 8 | keyword | 疑问句中的关键词（目前全是空） |
| 9 | type | 问题的类型 |
| 10 | answer\_type | 要求答案的类型 |
| 11 | time | 入库时间 |
| 12 | index | 建立索引时的标志位 |

**特别说明：**

**1: type，问题类型，即封闭式问题（CLOSE），开放式问题（OPEN）**

**2：answer\_type，答案类型，**

1. **药物 *M* = {"药","药物","吃","喝"};**
2. **饮食 *Y* = {"食物","饮食","营养","吃","喝"};**
3. **治疗 *T* = {"注意","措施","治","康复","预防","法"};**
4. **检查-指标 *I* = {"检查","诊断","标准"};**
5. **病因 *R* = {"原因","为什么","为何"};**
6. **危害 *W* = {"害","副作用","作用","效果","影响","危险"};**
7. **疾病描述 *D* = {"病","状态","情况","阶段"};**

**3：keyword，疑问句中的关键词，目前值全是空，查看代码也是设置为空，并未做处理。（估计就是为了记录疑问句中出现的表示药物、饮食、治疗、检查、病因、危害、疾病等的关键词。）**

**4：index，在建立索引时，如果对该问题创建了索引则值为1，如果没有则值为0.**

1. **answer表**

|  |  |  |
| --- | --- | --- |
| 序号 | 字段 | 字段描述 |
| 1 | Id | 对应reply\_post表中的id |
| 2 | Plength | 片段个数 |
| 3 | anno\_content | 标注后的答案内容 |
| 4 | Content | 答案内容 |
| 5 | question | 对应的问题id |
| 6 | Time | 入库时间 |
| 7 | Index | 建立索引时的标志位 |

1. **piece表**

|  |  |  |
| --- | --- | --- |
| 序号 | 字段 | 字段描述 |
| 1 | id | 片段id，自增，从1开始 |
| 2 | pos | 片段在答案中的位置，位置从0开始 |
| 3 | category | 片段分类（总分） |
| 4 | category\_2 | 片段分类（细分） |
| 5 | context | 片段是否依赖问题上下文 |
| 6 | anno\_content | 片段标注后的文本内容 |
| 7 | content | 片段文本内容 |
| 8 | question | 问题id |
| 9 | answer | 答案id |
| 10 | time | 入库时间 |
| 11 | index | 建立索引时的标志位 |

**特别说明：**

1. **category 表示片段的分类，从总的方面说明。代码中涉及的有{d, i, k, m, y, t, cq, fw, ab}，而数据库中主要有{ t、m、y、d、ab}，**

**其中，t表示一般的治疗手段；**

**m表示药物**

**y****表示食物**

**d****表示疾病或者症状的描述、内容等；**

**i****表示诊断标准或者检查项目**

**k?**

**cq?**

**fw?**

**ab?**

1. **category\_2 也是片段的分类，从细的方面划分：**

**{"d", "cd", "dr", "cdr", "dw", "cdw"}**

**{"i", "iz", "ciz", "iq"}**

**{"m", "cm", "mw"}**

**{"y", "cy", "yw"}**

**{"t", "ct", "tq"}**

**{"k"}**

**{"cq"}**

**{"fw"}**

**{"ab"}**

**红色标注的为在数据库中出现的类别。**

**（1）d，表示疾病或者症状的描述、内容等；**

**cd，依赖上下文构成完整语义**

**dr，导致疾病或者症状的原因**

**cdr，依赖上下文**

**dw，疾病或者症状带来的危害**

**cdw，依赖上下文**

**（2）i，表示诊断标准或者检查项目**

**iz，诊断标准或者检查项目的诊断标准**

**ciz，依赖上下文**

**iq，其他。例如：定期检查，全面检查，注意复查。。。**

**（3） m，表示药物**

**cm，依赖上下文**

**mw，药物治疗的效果或者副作用**

**（4）y， 表示食物**

**cy，依赖上下文**

**yw，饮食治疗的效果或者副作用**

**（5）t，一般的治疗手段**

**ct，依赖上下文**

**tq，其他。例如，不用担心；你这个不用太担心的。**

1. **context 片段是否依赖上下文**

# 详细流程

### 数据的来源与去向

**来源**：数据库ishc\_data中的post表和reply\_post表，处理函数package application.qav2.process\_data中Preprocess. fetch()方法。

**去向**：数据库healthqa中的question表，answer表，piece表，以及index目录下entitymap.db。

A．在形成以上三张表的过程中包括问题处理、答案处理和片段处理，处理函数Transform.*transform*(qa)。

1. 问题处理
2. 问题标注，处理函数**package** application.qav2.process\_question.entity中的QuestionEntity.*annotate*(qa.q.raw\_content)。
3. 实体抽取，处理函数**package** application.qav2.process\_data中的*extractEntities*(anno\_q)。
4. 结构化表示，处理函数**package** application.qav2.process\_question.structual中的QuestionParser.*structualRepresent*(anno\_q)。
5. 意图分析，包括抽取疑问词，问题分类（open/close），回答类型(M、*Y、T…..*)，处理函数QuestionIntention.*extractBody*(qa.q.raw\_content)。
6. 答案处理
7. 答案标注，处理函数**package** application.qav2.process\_answer.entity中的AnswerEntity.*annotate*(a.raw\_content)。
8. 最小完整语义切分，处理函数**package** application.qav2.process\_answer.wzyy\_crf中的Wzyy.*annotate*(anno\_content)。其中answer表中只是记录了答案经过最小完整语义切分后的片段数。
9. 答案实体抽取，处理函数**package** application.qav2.process\_data中*extractEntities*(anno\_content);
10. 片段处理
11. 片段分类和片段位置，处理代码在Transform.*transform*(qa)中，标注为“split answer into pieces and classification”
12. 片段实体抽取，处理函数**package** application.qav2.process\_data中*extractEntities*(piece);

B． entitymap.db保存的是问题和答案的实体，它的形成过程是Transform.*entity\_map*.toDB();

注：1，数据处理的过程重点看**package** application.qav2.process\_data中的Preprocess.java和Transform.java

2,entitymap.db保存在*sqlite*数据库中，可以使用doc目录下SQLiteDatabaseBrowserPortable.exe打开查看。

## QA后端代码执行过程

执行入口：**package** application.qav2中的Ask.ask()

### Find top questions

处理函数findTopQuestions( question\_str, 15, 0.5)，其中问题检索函数是QuestionSearcher.searchQuestion()，方法是lucene检索。

1. 如果检索出的问题中不包含输入的问题的实体，则过滤掉，反之，保留。
2. 如果问题的得分小于设置的阈值，则过滤，反之，保留。
3. 对获取的问题列表排序Collections.*sort*(temp);

### Find top answers

处理函数Summarization.*findTopAnswers*(question, top\_questions);

1，根据问题列表获取答案列表HealthqaDB.*getAnswerList*(ql);

2，计算答案得分RankAnswer.*findTopAnswers*(input\_question, questions, all\_answers);这个方法中记录了计算得分的指标，比如tfidf，QKW，AKW等等（你可以在此方法中加入其它指标计算答案得分）。

1. 排序Collections.*sort*(temp);

### 3.2.3 Find top pieces

处理函数Summarization.*findTopPieces* *(question, answers);*

1. 根据答案获取片段列表HealthqaDB.*getScoredPieceList*(answers);
2. 计算片段得分RankPiece.*findTopPieces*(input\_question, all\_answers);这个函数中记录了计算片段得分的指标，比如tfidf，pos，QKW等等（你可以在此方法中加入其它指标计算片段的得分）。
3. 排序Collections.*sort*(temp);
4. 去除相似度大于0.5的片段。

### 小结

最后我们把以上各个过程处理后的结果写入到tmp目录下的文件中，以输入问题的id号命名。如果发现QA系统的用户界面呈现的结果出现问题，可以查看此文件，然后跟踪到具体的处理函数查找原因。

## QA前段代码执行过程

#### 在app.js中规划相关的路由

首页：/

仅答案：/onlyAnswers

仅片段：/onlyPieces

片段组合：/combinePieces

#### 路由routes下的文件进行处理

onlyAnswers.js

onlyPieces.js

combinePieces.js

以上三个文件是我后来加上的，为了对应于页面上三个标签的处理，但是我个人比较懒，所以最终都在index.js中进行处理了，分别对应于/ask\_a 、 /ask\_p 、ask\_pz。

#### views下的文件显示对应的结果

onlyAnswers.jade 显示“仅答案”的内容

onlyPieces.jade 显示“仅片段”的内容

combinePieces.jade 显示“片段组合”的内容

index.jade 首页

error.jade 错误页面

layout.jade 公共定义，AngularJS

# 4 打包

在Eclipse中，项目右键—导出—java文件—runnable jar file