# 基于知识图谱的医疗知识可视化及智能问答系统

# 一、项目介绍

利用信息技术手段开展医疗知识的管理和服务是一项开创性的探索，在医学领域上具有极大的应用价值。知识图谱是近年来知识管理和知识服务领域中出现的一项新兴技术，因其简单易学、可扩展性强、支持智能应用等优点而得到广泛应用，它为医疗知识的关联、整合与分析提供了理想的技术手段，它有助于实现疾病、诊断检查项目、疾病症状以及药品知识等各类知识的关联与整合，挖掘整理以疾病为核心的相关知识，实现智能化、个性化的医疗知识服务，在医疗领域具有广阔的应用前景。

鉴于此，本小组基于医疗知识，初步建立了由疾病、诊断检查项目、症状、医疗科室、药品、食品等7个实体所构成的医疗知识图谱，不仅提供疾病的易发人群、持续时间、治愈概率、治疗方式、疾病原因、防御措施这些概念知识和医案文本以供参考，同时以图形化的方式呈现“疾病——药品”、“疾病——疾病症状”、“疾病——食品”、“疾病——医疗科室”等实体之间的相互关系，实现医疗知识体系可视化，以促进医疗知识的互融互通辅助医学研究和治疗决策。同时，本小组基于构建好的医疗知识图谱，实现了以传统规则匹配的智能问答，用户可快速找到与当前研究疾病（如症状、诊断检查项目、并发症、所属科室、推荐食谱、忌吃、宜吃、推荐药品等）相关的医案、指南和知识库内容，辅助用户进行决策。系统协助用户在概念层次上浏览医疗知识，发现概念或知识点之间的潜在联系，从而更好地驾驭复杂的医疗知识体系。

# 二、背景概述

知识图谱（Knowledge Graph），在图书情报界称为知识域可视化或知识领域映射地图，是显示知识发展进程与结构关系的一系列各种不同的图形，用可视化技术描述知识资源及其载体，挖掘、分析、 构建、绘制和显示知识及它们之间的相互联系。 知识图谱是通过将应用数学、 图形学、信息可视化技术、信息科学等学科的理论与方法与计量学引文分析、共现分析等方法结合，并利用可视化的图谱形象地展示学科的核心结构、发展历史、 前沿领域以及整体知识架构达到多学科融合目的的现代理论，能为学科研究提供切实的、有价值的参考。知识图谱，是结构化的语义知识库，用于以符号形式描述物理世界中的概念及其相互关系，其基本组成单位是“实体-关系-实体”三元组，以及实体及其相关属性-值对。实体之间通过关系相互联结，构成网状的知识结构，即语义网络（Semantic Network)。知识图谱在语义网络框架中填充了大量的知识内容。这些知识内容来自数据库、文献库、数据文件等各种数字化资源。知识图谱对分散的知识进行汇集和组织，可以有助于实现知识资源的关联与整合，为解决“知识孤岛”问题提供了理想的技术手段。

随着智能时代的到来，把临床数据、临床指南、医疗知识通过大数据、知识图谱、可视化系统结合，核心医学概念的全面覆盖、医疗生态圈内全方位知识数据的聚合，构建综合医疗大脑，给临床医生、科研工作者、管理工作者提供帮助，成为未来医疗的新兴发展方向。医学领域有其自身的特点和需求，需要专门研究医疗知识建模方法，解决医疗知识的获取、分类、表达、组织、存储等核心技术问题，采集加工高质量的医疗知识，挖掘实体之间的关系，才能建立准确、实用、完整的医疗知识图谱。

本研究以疾病为中心，对医学领域庞大的医疗疾病知识内容进行系统梳理，挖掘“疾病——药品”、“疾病——疾病症状”、“疾病——食品”、“疾病——医疗科室”等实体之间的关系，初步建立了一个医疗知识图谱系统。该系统以医学领域本体作为骨架，集成了疾病所需检查、疾病推荐食谱、疾病忌吃及宜吃、药品在售药品、疾病推荐药品等多种知识资源，并实现了各类知识点之间的知识关联。知识图谱为疾病知识体系的系统梳理和深度挖掘提供了新颖的方法，有助于实现医疗知识的关联、整合与可视化，促进疾病研究，辅助疾病诊断治疗。另外，基于知识图谱进一步实现医疗领域的智能问答，将知识图谱的内容以另一种形式呈现，方便用户自由检索。

# 三、数据获取

本项目立足医药领域，以中科院软件所github上的开源数据集为数据来源，如图1，剔除一些冗余数据后，最终以疾病为核心，选取了7类规模为4.4万的实体，9类规模约28万实体关系，8类疾病属性类型。

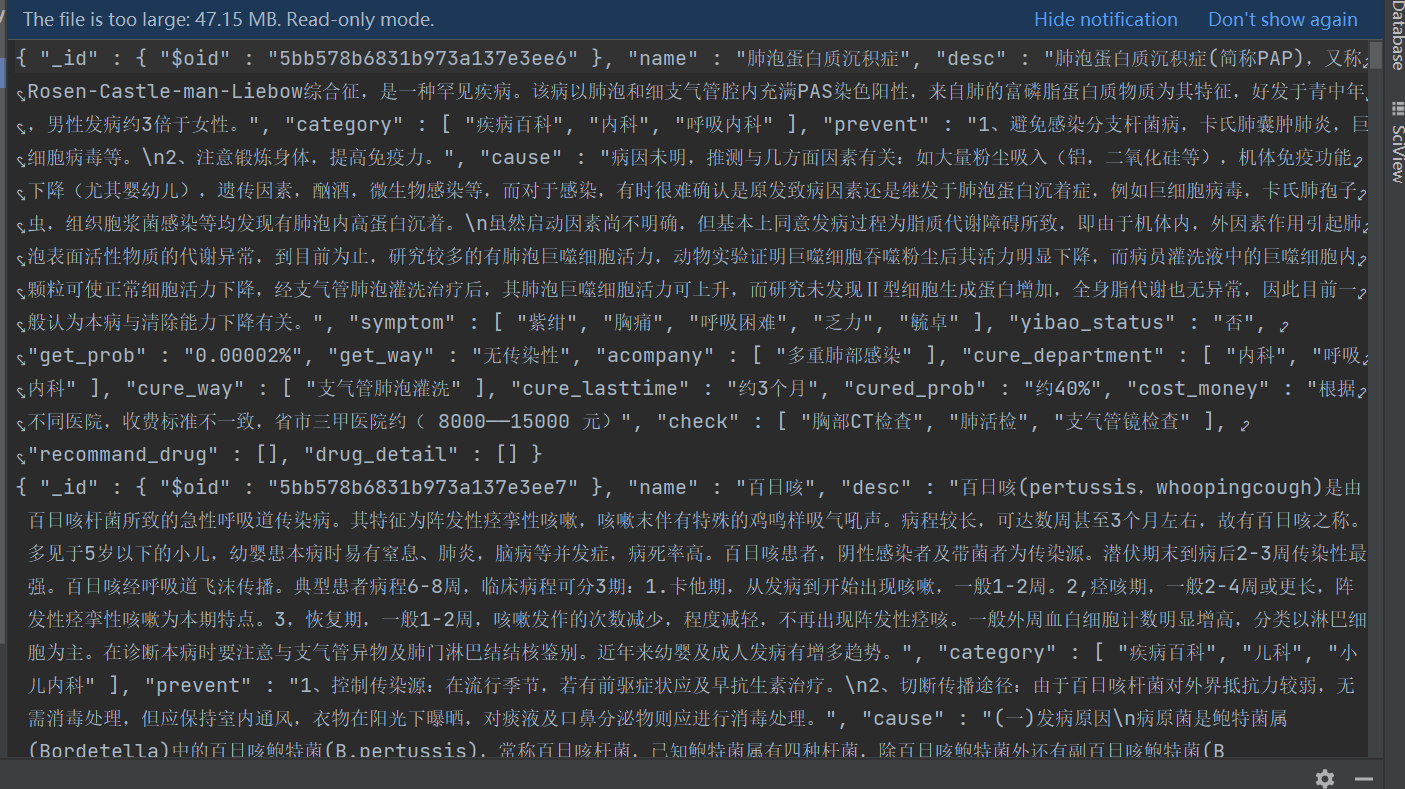


图 1 medical.json

## 1.实体类型

|  |  |  |  |
| --- | --- | --- | --- |
| **实体类型** | **中文含义** | **实体数量** | **举例** |
| Check | 诊断检查项目 | 3,353 | 支气管造影;关节镜检查 |
| Department | 医疗科室 | 54 | 整形美容科;烧伤科 |
| Disease | 疾病 | 8,807 | 血栓闭塞性脉管炎;胸降主动脉动脉瘤 |
| Drug | 药品 | 3,828 | 京万红痔疮膏;布林佐胺滴眼液 |
| Food | 食物 | 4,870 | 番茄冲菜牛肉丸汤;竹笋炖羊肉 |
| Producer | 在售药品 | 17,201 | 通药制药青霉素V钾片;青阳醋酸地塞米松片 |
| Symptom | 疾病症状 | 5,998 | 乳腺组织肥厚;脑实质深部出血 |
| Total | 总计 | 44,111 | 约4.4万实体量级 |

## 2.实体关系类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实体关系类型** | **中文含义** | **实体对应关系** | **关系数量** | **举例** |
| belongs\_to | 属于 | (Disease)-[belongs\_to]- (Department) | 8,844 | <妇科,属于,妇产科> |
| do\_eat | 疾病宜吃食物 | (Disease)-[do\_eat]-(Food) | 22,237 | <胸椎骨折,宜吃,黑鱼> |
| drugs\_of | 药品在售药品 | (Drug)-[drugs\_of]- (Producer) | 17,315 | <青霉素V钾片,在售,通药制药青霉素V钾片> |
| need\_check | 疾病所需检查 | (Disease)-[need\_check]- (Check) | 39,422 | <单侧肺气肿,所需检查,支气管造影> |
| no\_eat | 疾病忌吃食物 | (Disease)-[no\_eat]- (Food) | 22,246 | <唇病,忌吃,杏仁> |
| recommand\_drug | 疾病推荐药品 | (Disease)-[recommand\_drug]-(Drug) | 59,467 | <混合痔,推荐用药,京万红痔疮膏> |
| recommand\_eat | 疾病推荐食谱 | (Disease)-[recommand\_eat]- (Food) | 40,221 | <鞘膜积液,推荐食谱,番茄冲菜牛肉丸汤> |
| has\_symptom | 疾病症状 | (Disease)- [has\_symptom]- (Symptom) | 5,998 | <早期乳腺癌,疾病症状,乳腺组织肥厚> |
| acompany\_with | 疾病并发疾病 | (Disease)-[acompany\_with]- (Disease) | 12,029 | <下肢交通静脉瓣膜关闭不全,并发疾病,血栓闭塞性脉管炎> |
| Total | 总计 | —— | 279,498 | 约28万关系量级 |

## 3.实体属性类型

|  |  |  |
| --- | --- | --- |
| 属性类型 | 中文含义 | 举例 |
| name | 疾病名称 | 喘息样支气管炎 |
| desc | 疾病简介 | 又称哮喘性支气管炎... |
| cause | 疾病病因 | 常见的有合胞病毒等... |
| prevent | 预防措施 | 注意家族与患儿自身过敏史... |
| cure\_lasttime | 治疗周期 | 6-12个月 |
| cure\_way | 治疗方式 | "药物治疗","支持性治疗" |
| cured\_prob | 治愈概率 | 95% |
| easy\_get | 疾病易感人群 | 无特定的人群 |

# 四、运行环境

* python：3.0及以上
* neo4j ：4.1.11
* jdk：15.0.2
* flask：2.0.2

# 五、项目结构

├── README.md
  
├── \_\_pycache\_\_ \\编译结果保存目录
  
│ ├── answer\_search.cpython-36.pyc
  
| ├── chatbot\_graph.cpython-36.pyc
  
│ ├── question\_classifier.cpython-36.pyc
  
│ └── question\_parser.cpython-36.pyc
  
├── app.py \\整个系统的主入口
  
├── data
  
│ └── medicaln.json \\本项目的全部数据，通过build\_medicalgraph.py导入Neo4j
  
├── dict
  
│ ├── check.txt \\诊断检查项目实体库
  
│ ├── deny.txt \\否定词库
  
│ ├── department.txt \\医疗科目实体库
  
│ ├── disease.txt \\疾病实体库
  
│ ├── drug.txt \\药品实体库
  
│ ├── food.txt \\食物实体库
  
│ ├── producer.txt \\在售药品库
  
│ └── symptom.txt \\疾病症状实体库
  
├── inteligent\_QA \\智能问答系统模块
  
| ├── chatbot\_graph.py \\问答主程序
  
│ ├── question\_classifier.py \\根据特征词对问句类型进行分类   
│ ├── question\_parser.py \\问句解析脚本
  
│ └── answer\_search.py \\执行cypher查询，并返回相应结果匹配回答模板
  
├── neo\_db \\知识图谱构建模块
  
│ ├──config.py \\配置连接Neo4j参数以及系统默认参数
  
│ ├──build\_medicalgraph.py \\创建知识图谱，图数据库的建立
  
│ ├──query\_graph.py \\知识图谱的查询
  
├── templates \\HTML的页面
  
│ ├── index.html \\欢迎界面
  
│ ├── all\_relation.html \\知识图谱及问答系统页面
  
├── static \\用以设置页面的样式和动态效果
  
│ |-css \\设置页面的样式
  
│ ├── js \\存放echarts、jquery等系统中用到的开源库  
│ ├── fonts \\存放系统中所需字体样式
  
│ ├── images \\存放系统所用的图片

# 六、技术架构

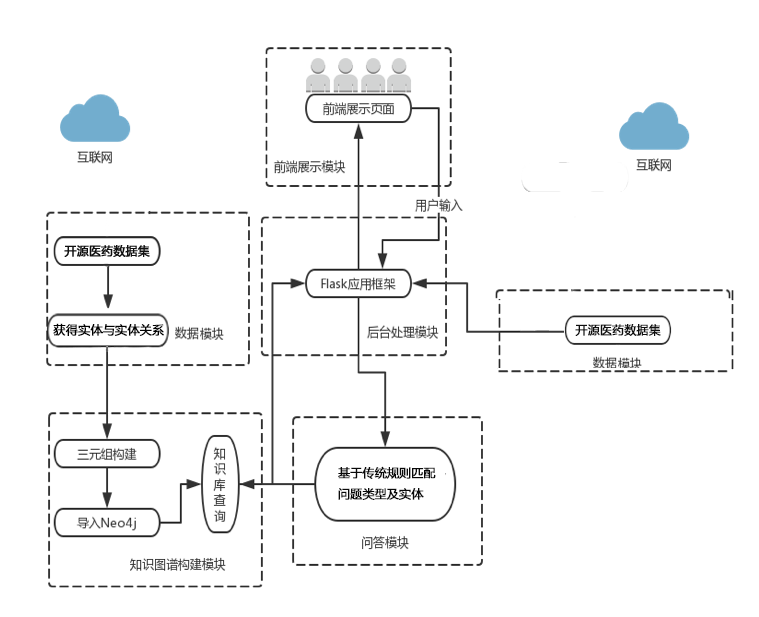


图 2 整体技术架构图

如图2系统整体技术架构图所示：

1.获取开源医药数据集数据，获得实体与实体关系。

2.根据实体和实体关系构建三元组，然后将其导入Neo4j，生成知识图谱。

3.根据构建好的医药知识图谱，基于传统规则匹配问题类型与实体，在知识图谱中查询结果后组装回答返回，实现智能问答。

4.通过flask web搭建服务，使用echarts的力引导图布局对知识图谱进行渲染，从而实现在本地系统进行可视化展示。

# 七、部署步骤

1. 配置要求：要求配置Neo4j数据库及相应的python依赖包。启动Neo4j 数据库服务器Neo4j数据库用户名密码记住，并修改相应文件。

2. 知识图谱数据导入：python build\_medicalgraph.py，导入的数据较多，估计需要几个小时。

3. 启动智能问答：python chat\_graph.py

4. 启动本地系统：app.py

# 八、医药知识图谱构建

## （一）Neo4j

知识图谱由于其数据包含实体、属性、关系等，常见的关系型数据库诸如MySQL之类不能很好的体现数据的这些特点，因此知识图谱数据的存储一般是采用图数据库（Graph Databases）。而Neo4j是其中最为常见的图数据库。Neo4j是一个高性能的NoSQL图形数据库，它的数据并非保存在表或集合中，而是保存为节点以及节点之间的关系。

知识图谱是一种基于图的数据结构，由节点和边组成。其中节点即实体，由一个全局唯一的 ID 标示，关系即边用于连接两个节点。通俗地讲，知识图谱就是把所有不同种类的信息连接在一起而得到一个关系网络，提供了从“关系”的角度去分析问题的能力。而 Neo4j 作为一种经过特别优化的图形数据库，有以下优势：

* 拥有简单的查询语言 Neo4j CQL
* 遵循属性图数据模型
* 通过使用 Apache Lucence 支持索引
* 支持 UNIQUE 约束
* 包含一个用于执行 CQL 命令的 UI：Neo4j 数据浏览器
* 支持完整的 ACID（原子性，一致性，隔离性和持久性）规则
* 采用原生图形库与本地 GPE（图形处理引擎）
* 支持查询的数据导出到 Json 和 XLS 格式
* 提供了 REST API，可以被任何编程语言（如 Java，Spring，Scala 等）访问
* 提供了可以通过任何 UI MVC 框架（如 Node.js）访问的 Java 脚本
* 支持两种 Java API：Cypher API 和 Native Java API 来开发 Java 应用程序

### 1. Neo4j数据模型

* Node：节点，或者称为实体，包括标签、属性、属性值、id。
* Relationship：关系，包括标签、属性、属性值、id。关系是有方向的，（） - []→（）代表从左侧节点到右侧节点，创建关系必须指定方向，查询匹配关系可以不指定方向如下（） - [] - （），一个节点可以有一个关系是指向自己的。
* Property：属性，用key-value形式的键值对表示一个属性。节点和关系都可以设置属性。Property Key即属性的键。
* 每一个节点、关系都有一个唯一的id。

### 2. Cypher查询语言

Cypher对于Neo4j数据库就相当于SQL对于Mysql数据库，Cypher是Neo4j的查询语言。Cypher是专门为图像数据库设计的语言，允许用户不必编写图形结构的遍历代码，就可以对图形数据进行高效的查询。其具备的能力包括：对节点、关系进行增删改查、管理索引和约束等。例：查询属于内科的疾病如图3：

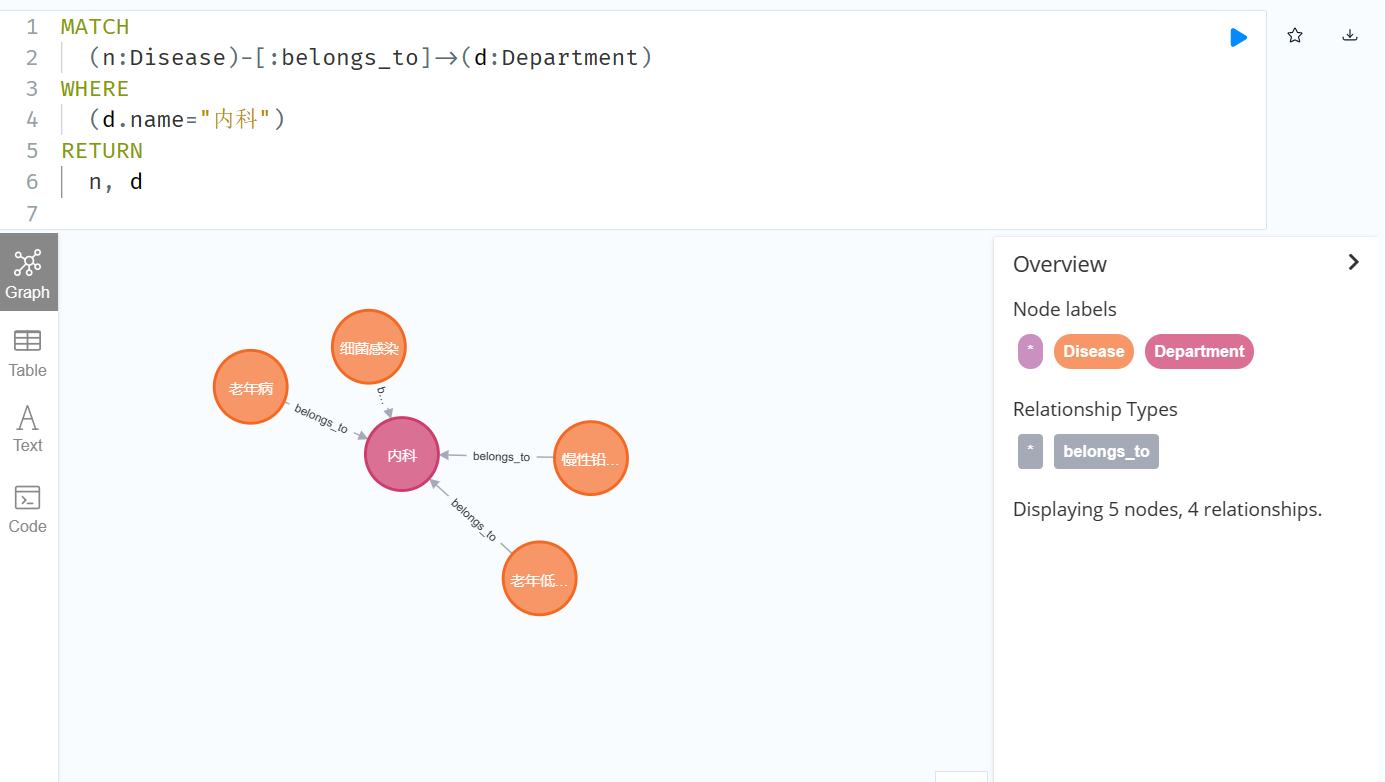


图 3 查询属于内科的疾病

### Py2Neo

Py2Neo是用来对接 Neo4j 的 Python 库，可通过Python应用程序内部和命令行使用Neo4j。在Py2Neo.database 模块中包含了和 Neo4j 数据交互的 API，最重要的当属Graph，它代表了 Neo4j 的图数据库，同时 Graph 也提供了许多方法来操作Neo4j 数据库。

## （二）构建知识图谱流程

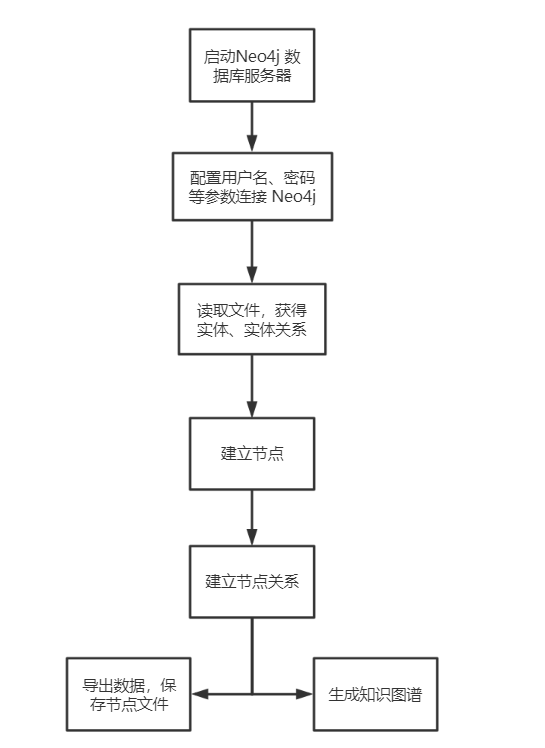
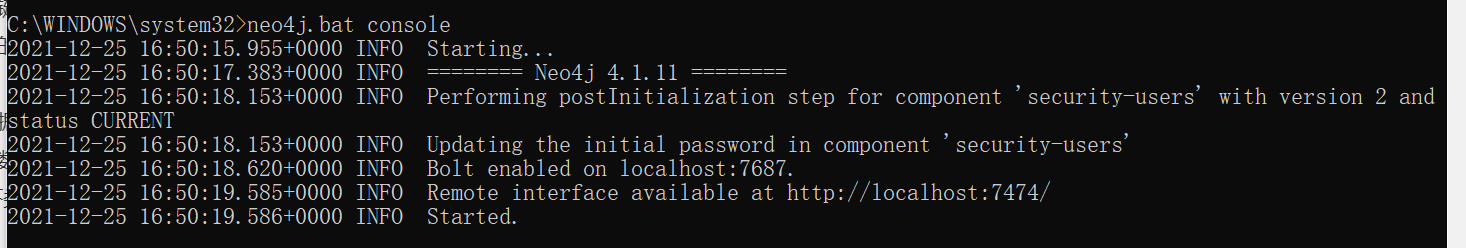
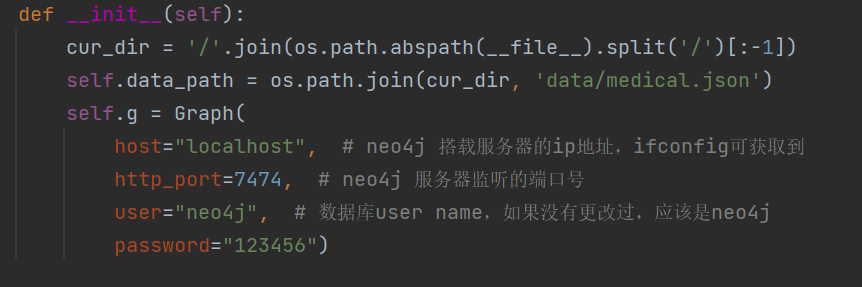


图 4 知识图谱构建流程图

### 1.启动Neo4j数据库服务器

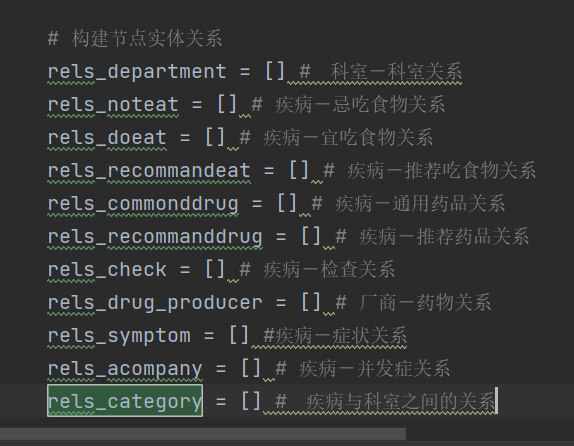


### 2.配置ip地址、端口、用户名、密码连接 Neo4j



### 3.读取文件，获得实体、实体关系

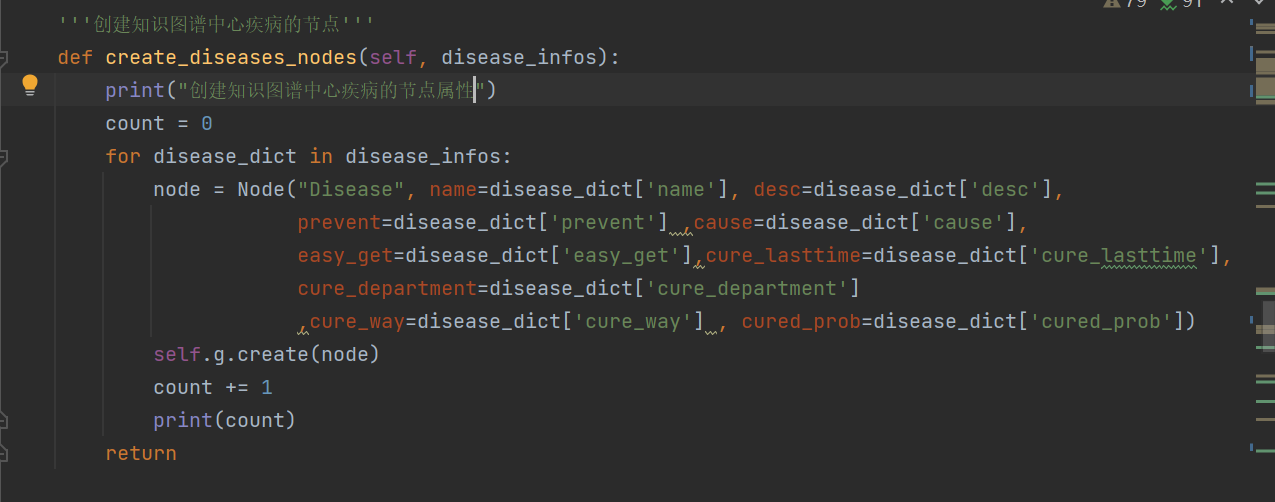




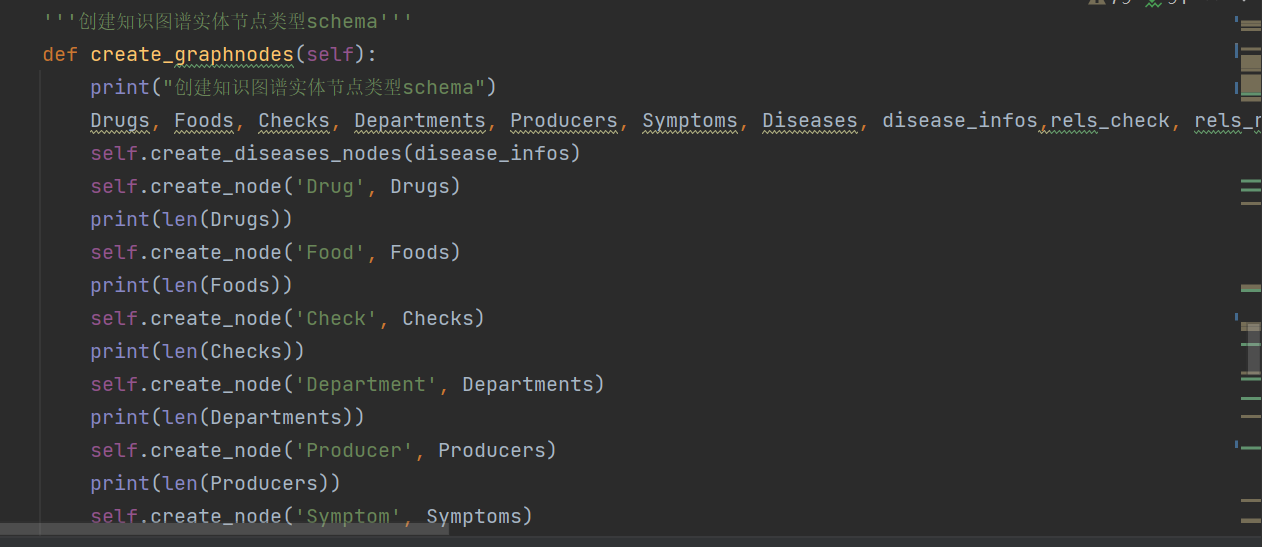
### 4.建立节点

根据医学领域知识，围绕疾病，共设计7类节点（Disease、Drug、Food、Check、Department、Producer、Symptom）。

（1）创建节点的属性：将每个节点的详细信息（如：疾病的易感人群、持续时间、治愈概率、治疗方式、疾病原因、防御措施）添加进去。



（2）创建实体节点：依照各个不同的标签（如：症状、部位...）创建节点名称。



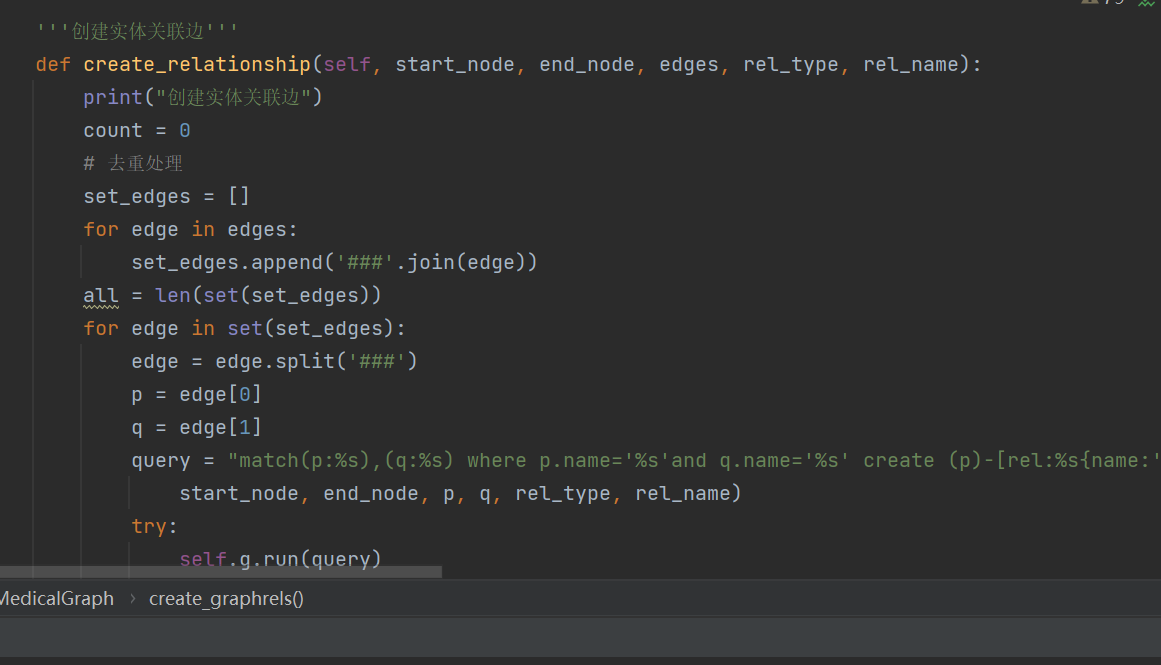
### 5.建立节点关系

根据医学领域知识，围绕疾病，共设计11种关系（Disease-(recommand\_eat)->Food，Disease-(no\_eat)->Food，Disease-(do\_eat)->Food，Department-(belongs\_to)->Department，Disease-(common\_drug)->Drug，Disease-(recommand\_drug)->Drug，Producer-(drugs\_of)->Drug，Disease-(need\_check)->Check，Disease-(has\_symptom)->Symptom，Disease-(acompany\_with)->Disease，Disease-(belongs\_to)->Department）。

通过之前整理好的疾病与每一个种类关系，建立起节点之间的关系，建立关系需要以下两个步骤：

（1）获得实体之间的关系

（2）通过Cypher查询节点将关系建立起来



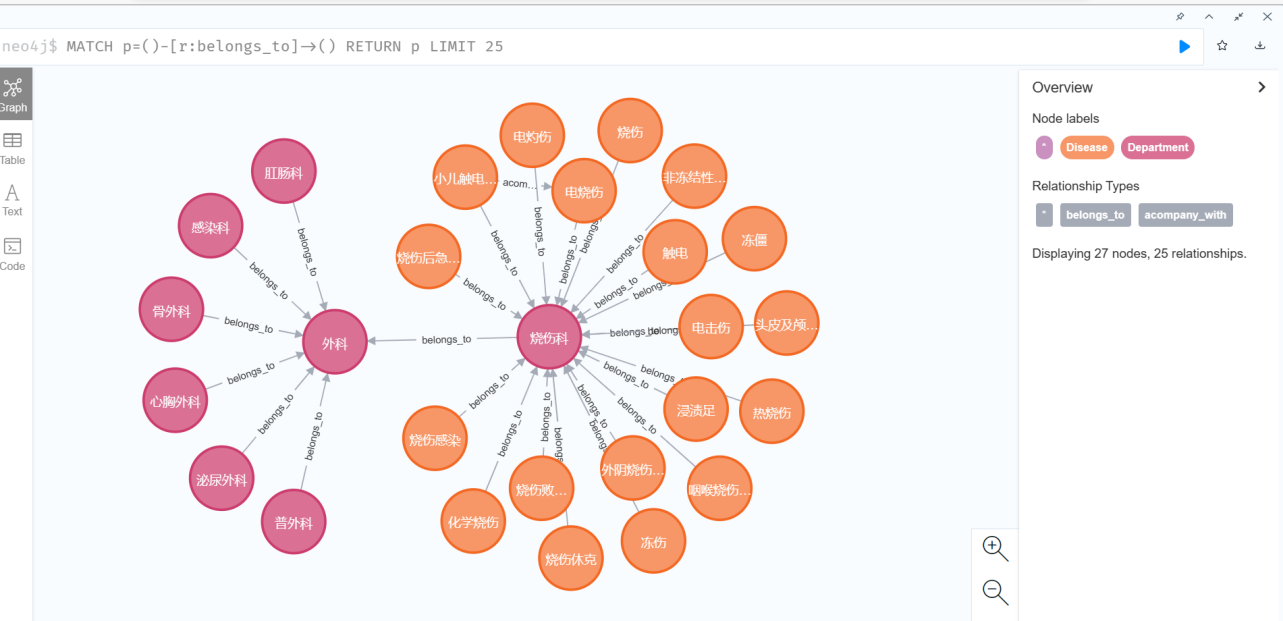
### 6.导出数据，保存节点文件

调用read\_nodes函数得到存储实体和实体间关系的变量，并逐行写入各变量对应的txt。



### 7.生成知识图谱

建立完成的图谱可以在浏览器中输入 [http://localhost:7474](https://link.zhihu.com/?target=http%3A//localhost%3A7474)，打开Neo4j可以查到建立完成的知识图谱。



# 九、基于医疗知识图谱的智能问答

本项目的问答系统依托已构建好的医药知识图谱，完全基于规则匹配实现，通过关键词匹配，对问句进行分类，医疗问题本身属于封闭域类场景，对领域问题进行穷举并分类，然后使用cypher的match去匹配查找Neo4j，根据返回数据组装问句回答，最后返回结果。

1. 依据特征词进行分类
2. 基于分类进一步分析，并生成查询语句
3. 基于查询语句在neo4j进行查询，并对返回的数据进行处理

## （一）技术架构

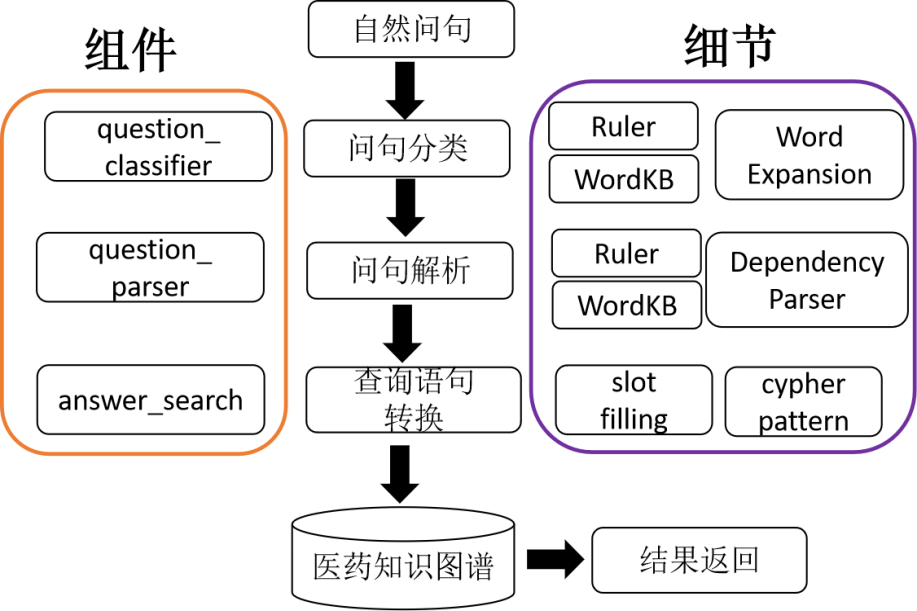


图 5 智能问答技术架构图

## （二）脚本结构

chatbot\_graph.py：问答程序脚本

question\_classifier.py：问句类型分类脚本

question\_parser.py：问句解析脚本

answer\_search.py ：问题查询脚本

## （三）实现智能问答流程

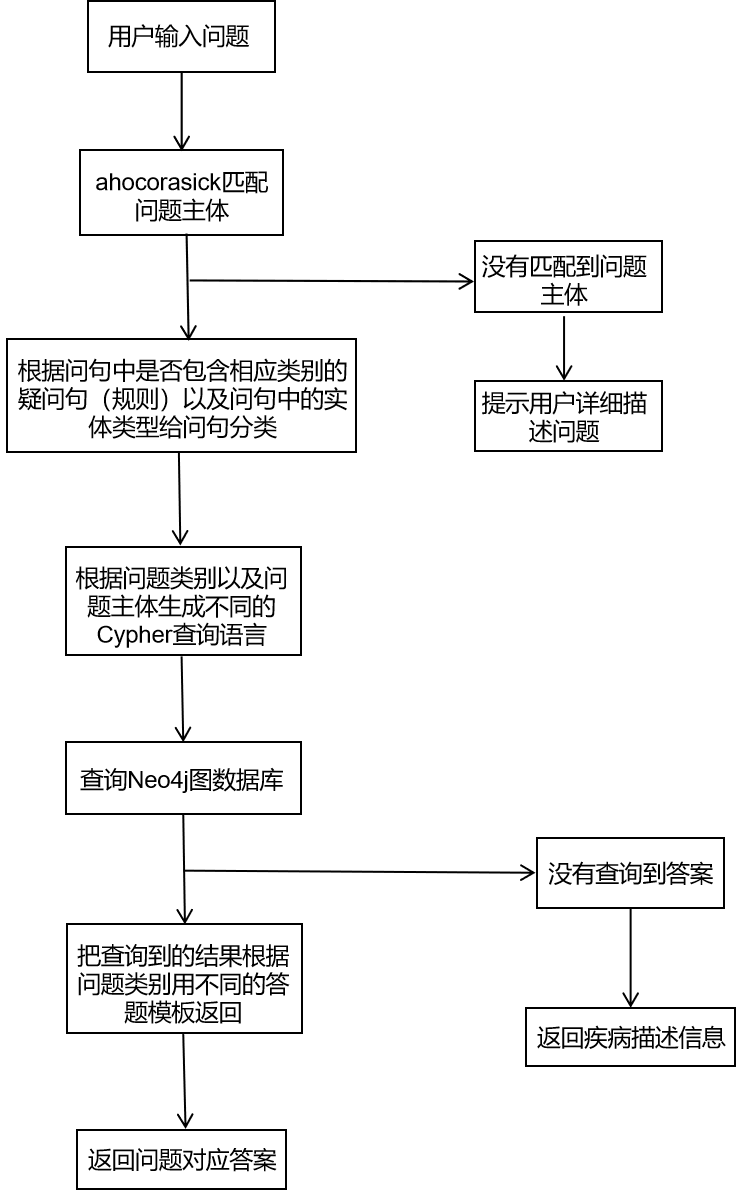


图 6 智能问答实现流程图

### 1.匹配用户问题主体，给问题分类

在自然语言处理过程中，最为关键的步骤就是实体提取和意图分类。中文与英文相比，中文词语之间缺少空格的分割，不太容易去区分词与词的界限，从而加大了自然语言处理对语句分割、语义理解上的困难。ahocorasick是一种字符串匹配算法，由两种数据结构实现：trie和Aho-Corasick自动机。Trie是一个字符串索引的词典，检索相关项时时间和字符串长度成正比。Aho-Corasick自动机（简称AC自动机）能够在一次运行中找到给定集合所有字符串。AC自动机通过将模式串构成Trie树，将主串匹配的过程变为在 Trie 树上转移的过程，可以完成多模式串的匹配。AC自动机状态转移规则放在Trie树上进行搜索匹配，能对目标串进行高效式搜索，增大了主串和模式串匹配的可能性，可以有效地提高处理问句的效率，因此本研究决定基于AC自动机算法处理用户输入的问题，利用知识图谱中存在的实体名字构成特征库（语料库）建立AC树（Aho-Corasick Tree），利用AC自动机算法，匹配问句中是否存在特征词，即查询问句中是否存在知识图谱实体名字，来实现实体提取。AC自动机用于在输入的一串字符串中匹配有限组“字典”中的子串。它与普通字符串匹配的不同点在于同时与所有字典串进行匹配，大大提升了匹配效率。

用户输入问题后，调用python的ahocorasick库对问题中的问句疑问词及实体特征词进行匹配，若没有匹配到相应的规则，则会提示用户重新详细描述问题，如图7。若匹配到相应类别的疑问句规则以及问句中的实体类别（如问疾病、症状、药品等），则根据匹配到的结果对输入的问句进行分类，构造出对应的 {特征词：特征词对应类型} 词典，从而确定问题类型以及问题主体。

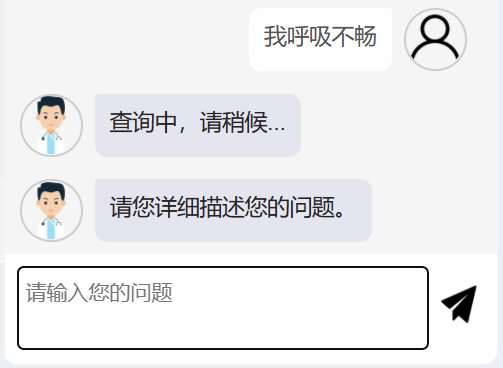


图 7 没有匹配到对应问题

### 2.问句解析

问句分类后还需对问句进行解析，获取到问句中领域词及其实体类型后，根据问题类别以及问题主体生成不同的 Cypher查询语言，便于后续neo4j图数据库查询结果。

### 3.查询Neo4j图数据库，根据返回的数据组装回答

执行cypher查询Neo4j图数据库，查询到对应问题的答案则返回相应结果，若没有查询到对应问题类型的答案，则查询出用户所提及的疾病或症状的描述信息返回，如图8，然后根据对应的问题类型，调用相应的回复模板对数据库查询返回的结果进行包装，最后返回答案给用户。

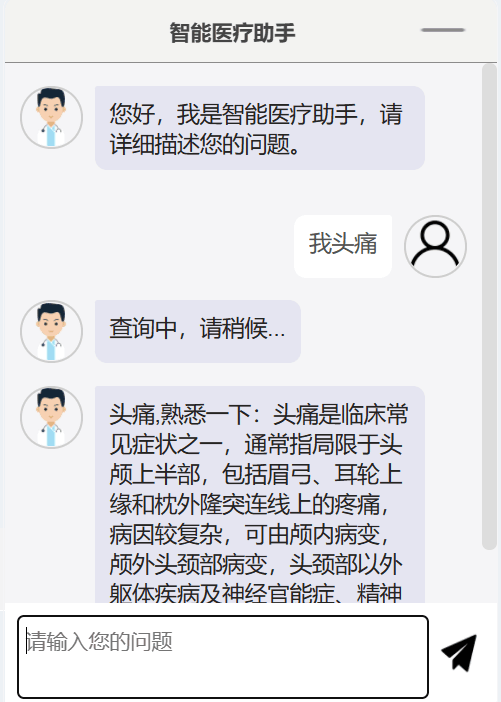


图 8 没有查询到对应问题类型的答案

## 支持问答类型

|  |  |  |
| --- | --- | --- |
| **问句类型** | **中文含义** | **问句举例** |
| disease\_symptom | 疾病症状 | 乳腺癌的症状有哪些？ |
| symptom\_disease | 已知症状找可能疾病 | 最近老流鼻涕怎么办？ |
| disease\_cause | 疾病病因 | 为什么有的人会失眠？ |
| disease\_acompany | 疾病的并发症 | 失眠有哪些并发症？ |
| disease\_not\_food | 疾病需要忌口的食物 | 失眠的人不要吃啥？ |
| disease\_do\_food | 疾病建议吃什么食物 | 耳鸣了吃点啥？ |
| food\_not\_disease | 什么病最好不要吃某事物 | 哪些人最好不好吃蜂蜜？ |
| food\_do\_disease | 食物对什么病有好处 | 鹅肉有什么好处？ |
| disease\_drug | 啥病要吃啥药 | 肝病要吃啥药？ |
| drug\_disease | 药品能治啥病 | 板蓝根颗粒能治啥病？ |
| disease\_check | 疾病需要做什么检查 | 脑膜炎怎么才能查出来？ |
| check\_disease | 检查能查什么病 | 全血细胞计数能查出啥来？ |
| disease\_prevent | 预防措施 | 怎样才能预防肾虚？ |
| disease\_lasttime | 治疗周期 | 感冒要多久才能好？ |
| disease\_cureway | 治疗方式 | 高血压要怎么治？ |
| disease\_cureprob | 治愈概率 | 白血病能治好吗？ |
| disease\_easyget | 疾病易感人群 | 什么人容易得高血压？ |

# 十、可视化展现

## （一）Flask Web服务搭建

Flask 是一个微型的 Python 开发的 Web 框架，基于 Werkzeug WSGI 工具箱和Jinja2模板引擎。Flask使用BSD授权。Flask 也被称为 “microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。然而，Flask 保留了扩增的弹性，可以用 Flask-extension 加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术等。

如图9所示，Flask的基本工作模式为：在程序里将一个函数分配给一个URL（Uniform Resource Locator，每一个信息资源在网上具有的唯一地址），每当有用户访问这个URL并发送请求时，系统就会执行此URL预分配好的函数，同时获取函数的返回值并将其显示到浏览器上。

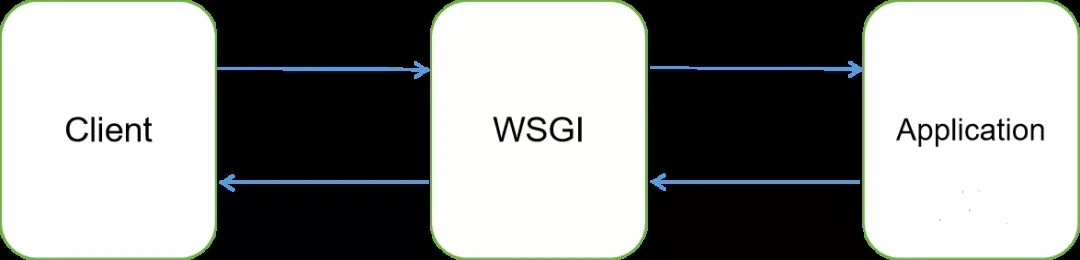


图 9 Flask的工作模式图

## **（二）ECharts力引导布局图**

ECharts，缩写来自 Enterprise Charts，商业级数据图表，是百度的一个开源的数据可视化工具，一个纯 Javascript的图表库，涵盖各行业图表，满足各种需求，能够在 PC 端和移动设备上流畅运行，兼容当前绝大部分浏览器（IE6/7/8/9/10/11，chrome，firefox，Safari等），底层依赖轻量级的 Canvas 库 ZRender，ECharts 提供直观，生动，可交互，可高度个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。

ECharts的力引导布局是模拟弹簧电荷模型在每两个节点之间添加一个斥力，每条边的两个节点之间添加一个引力，每次迭代节点会在各个斥力和引力的作用下移动位置，多次迭代后节点会静止在一个受力平衡的位置，达到整个模型的能量最小化。仿制neo4j进行数据展示，可以采用echarts力引导布局图实现，如图10。

实现原理：

1. echarts对象的初始化
2. 设置数据的类别
3. 编写功能配置信息option:标题、提示框、工具箱、图的类型、给节点和边赋值
4. 将option赋给echarts对象

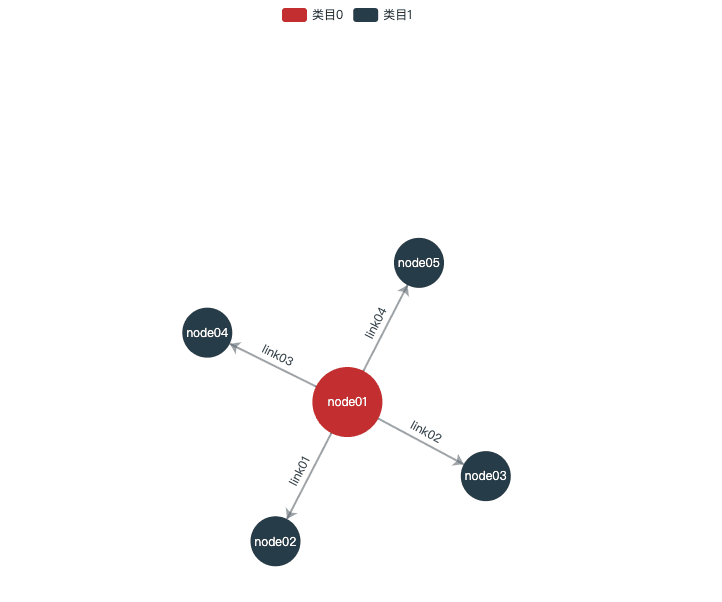


图 10 echarts力引导布局图

图 11

# 十一、项目最终效果

1.欢迎界面：用户点击“开启探索”的按钮，即可跳转到系统主界面



图 12 欢迎界面

2.系统主页面：系统主页面包括一个旋转标签云，将日常较为常见的疾病以标签的形式直观展示，用户可直接点击某个疾病标签，右侧将会生成对应疾病的知识图谱，展示出疾病的所属科室、并发症、症状、诊断检查、好评药品、忌吃、推荐食谱以及宜吃的详细信息。

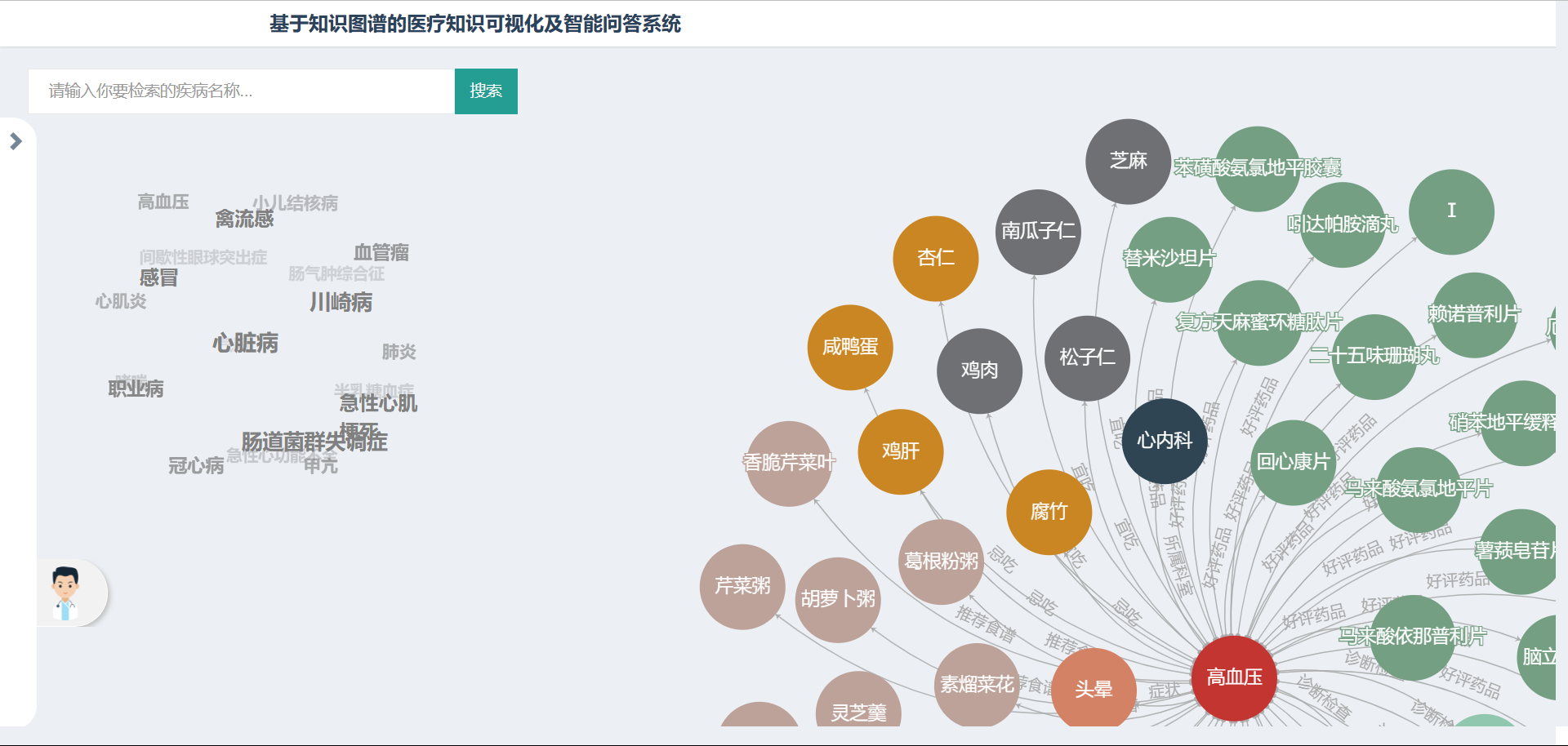


图 13 系统主界面

如图14，在生成的知识图谱中，疾病为中心节点，右侧的图例展现的是与疾病相关的项目，用户可任意选择隐藏不关注的方面，只呈现想要特殊关注的部分。

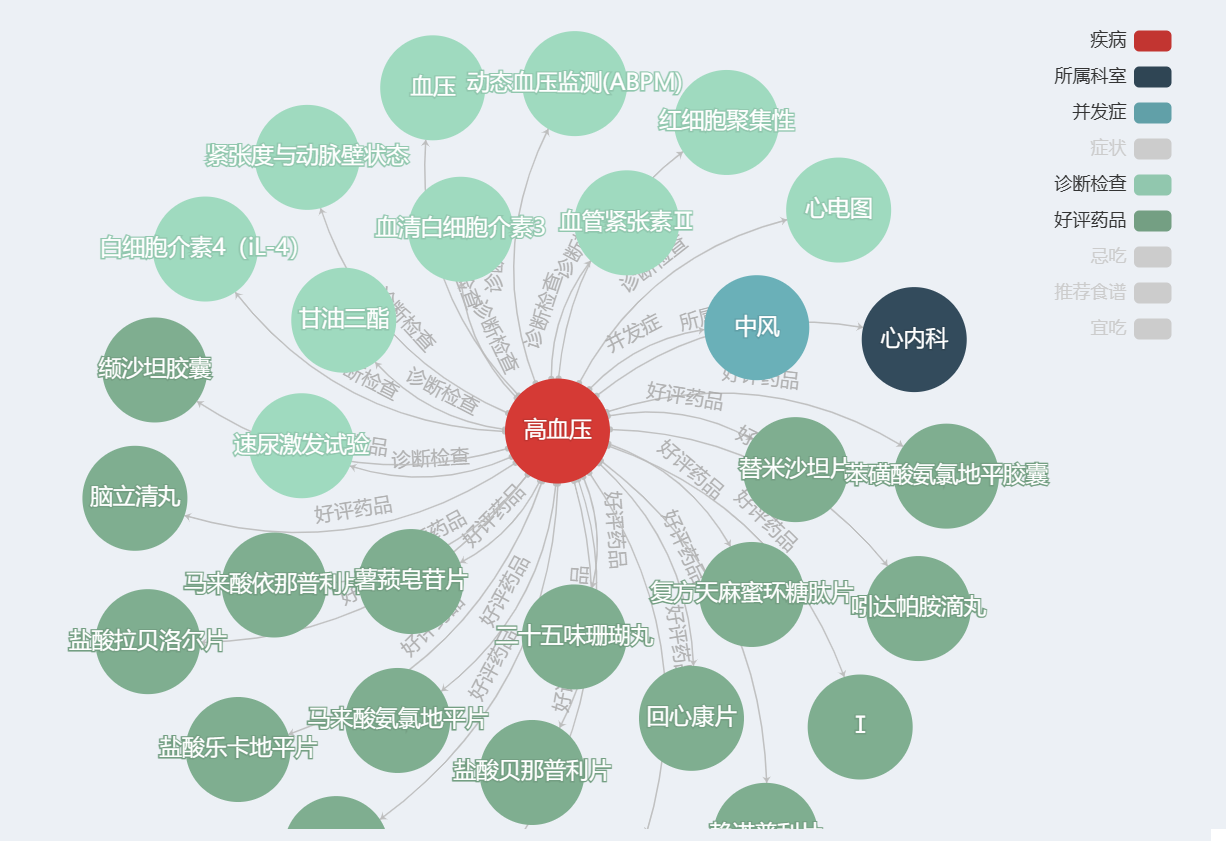


图 14 知识图谱可视化

如图15，展开左侧侧边栏，用户可以看到与该疾病相关的属性资料，主要包含疾病的易发人群、持续时间、治愈概率、治疗方式、相关介绍以及防御措施。

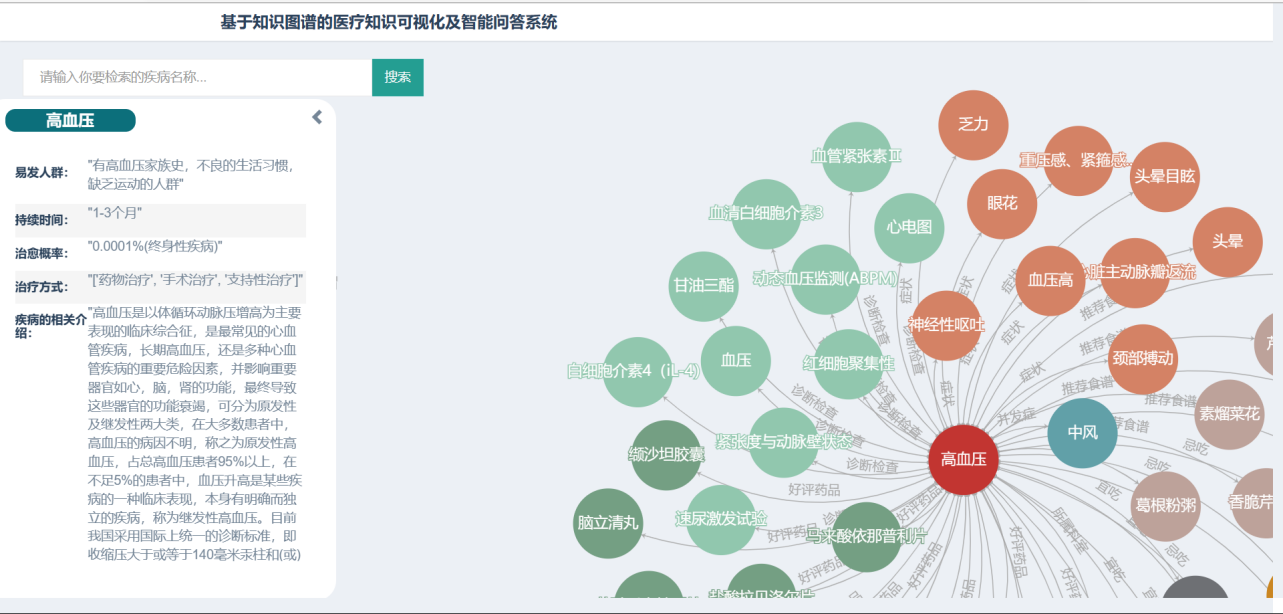


图 15 查看疾病的属性信息

如图16，用户也可顶部的搜索框直接检索某种疾病，输入需要检索的疾病名称，点击搜索后，即会生成对应的知识图谱，左侧侧边栏里的信息也会相应变换。

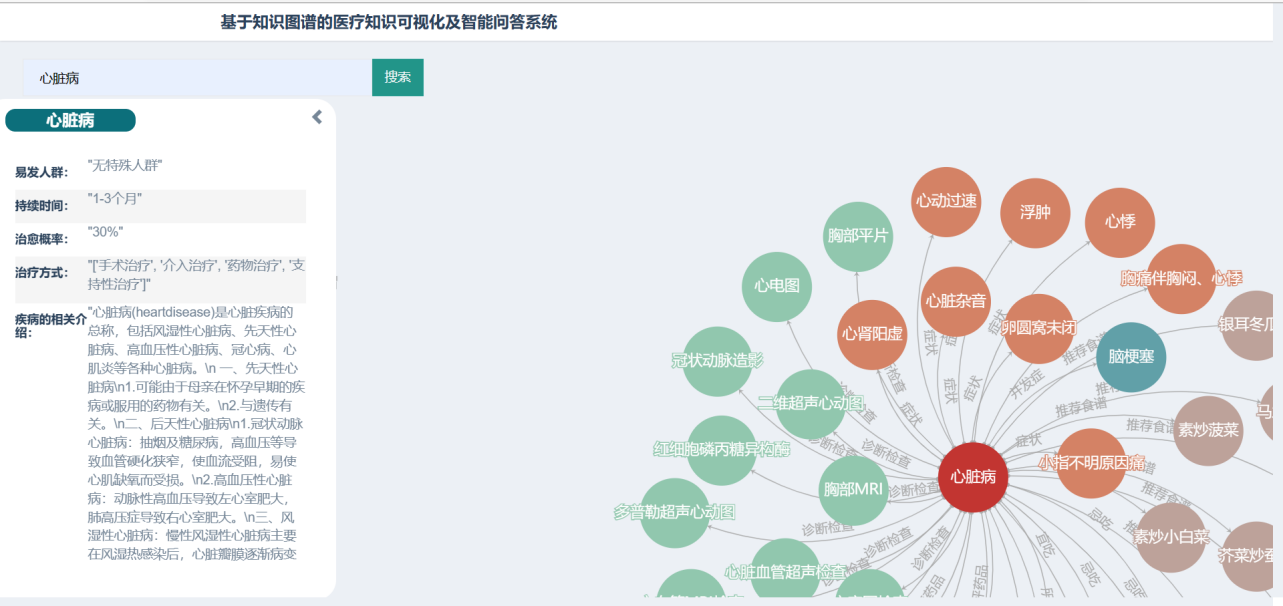


图 16 搜索

3.智能问答组件：如图17，点击左下角的医生图标，则会弹出智能医疗助手对话框，用户可在对话框中输入问题，智能医疗助手查询后会即刻返回响应结果。



图 17 智能问答

# 十二、总结与展望

本项目以Neo4j作为存储，立足医药领域，利用丰富的数据源，以疾病为核心，选取了7类规模为4.4万的实体，9类规模约28万实体关系，8类疾病属性类型，成功构建医疗知识图谱，不仅可获得疾病的治愈概率、疾病原因等属性信息，还可通过视觉的图形来感知“疾病——诊断检查项目”、“疾病——药品”、“疾病——食物”等实体之间的相关性，并基于flask web和echarts力引导布局图完成了医疗知识图谱在本地系统中的可视化。另外，本小组还基于传统规则的方式完成了知识问答，并最终以cypher查询语句作为问答搜索sql，在neo4j图数据库中查询到所需结果后返回，然后进一步用答复模板包装输出给用户，支持了问答服务。互联网诊疗可以使病人不出家门、不挂号就进行基础病情的诊断，具有很高的实用价值，而医疗知识图谱的构建以及智能医疗问答有助于对医疗知识进行分类整理和规范化表达，促进医疗知识的共享、传播与利用，在临床诊疗、临床研究、教育、培训等方面都具有应用价值。

本项目依旧存在不足：关于疾病的起因、预防等，实际返回的是一大段文字，这里其实可以引入事件抽取的概念，进一步将原因结构化表示出来。这个可以后面进行尝试。另外，在智能问答板块中，用户输入问题，现有系统只能基于给定的规则匹配出用户的问题类型，进而转换成相应的cypher语句去查询neo4j图数据库，然而有限的规则难以将变化无穷的实体和问题类型较全面地识别出来，难以涵盖所有的语言现象，因此特别容易产生错误，系统可移植性差。若这里可以考虑在用户输入问题后，系统可以对用户输入的问题进行语义匹配，引入faiss对句子的相似度进行匹配，那么系统查询的性能将更高效。后续我们将持续改进。