软件功能总体报告

该项目《中医艾灸可视化知识服务平台》主要分为以下若干栏目功能：

1.项目主页

2.条文详情

3.关键词搜索

4.知识图谱

本项目的知识图谱主要使用Neo4j数据库作为数据存储，数据关系来源于《实用常见病艾灸疗法》、《零基础学中医艾灸》、《艾灸穴位新解》、《简易针灸治疗法(艾灸篇)》等相关艾灸书籍。

4.1 关系抽取

本项目的关系抽取主要由以下步骤进行：

- 团队成员整理当前数据集，挑选出有代表性的书籍资料源，并搜集图片形式的书籍数据
- 团队成员协商确定过滤/筛选规则，针对特殊词汇进行注明，确定同义词（例如痛经在大部分书籍中表示为经痛，两者含义本质是相同的）
- 通过tesseract-OCR库对图片形式的书籍进行范围扫描，并且利用jieba分词库对OCR结果进行分词
- 提取分词结果中的实体，与已经整理好的匹配规则进行比对，整理形成病症-灸法-穴位的三元组
- 团队成员对抽取结果进行人工审查，确定抽取结果正确性，针对没有涉及到的部分情况进行补充。

通过上述方式，共筛选出三元关系1244条

同时，我们收集整理了病症、灸法、穴位与各自类别之间的联系，并将其同样整理到关系文件中。

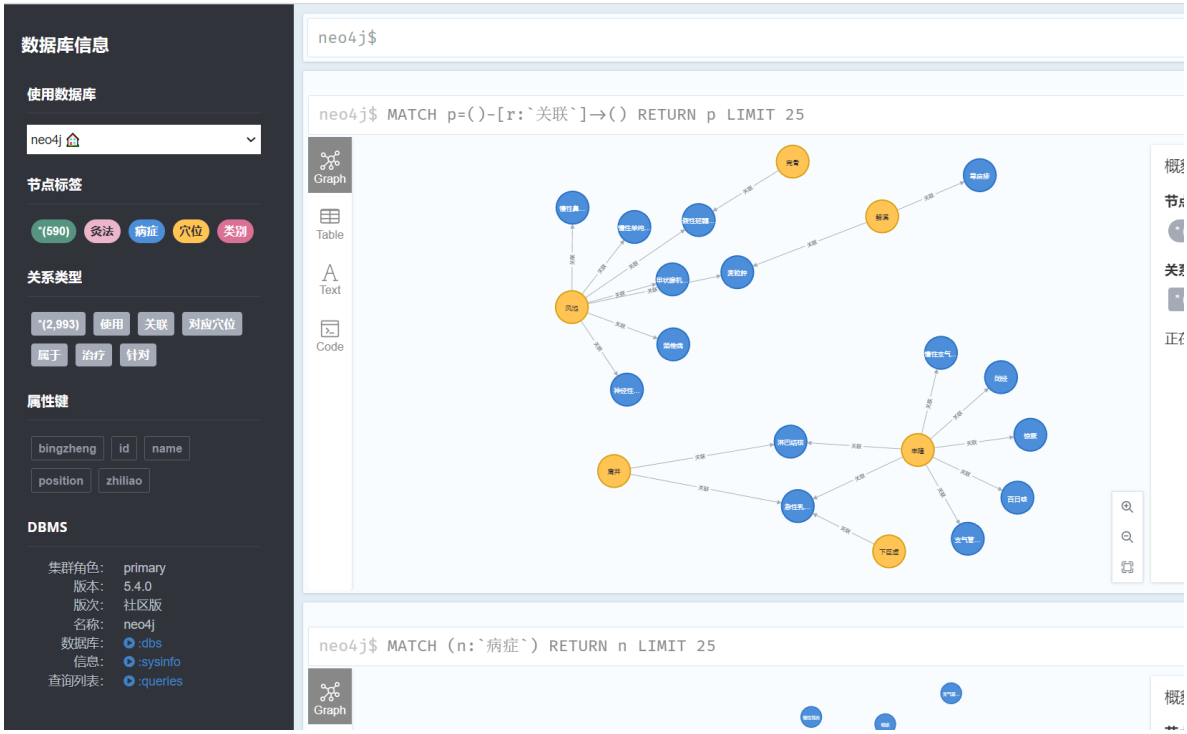
4.2 数据存储

> 对应文件：后端/graphGenerated.py

本项目的关系提取结果首先通过Excel表的形式存储

	A	B	C
1	病症	灸法	穴位
2	普通感冒	温和灸	大椎
3	普通感冒	温和灸	合谷
4	普通感冒	温和灸	列缺
5	普通感冒	温和灸	肺俞
6	普通感冒	雀啄灸	曲池
7	普通感冒	雀啄灸	合谷
8	急性黄疸型肝炎	温和灸	肝俞
9	急性黄疸型肝炎	温和灸	胆俞
10	急性黄疸型肝炎	温和灸	至阳
11	急性黄疸型肝炎	温和灸	阳陵泉
12	急性黄疸型肝炎	温和灸	阴陵泉
13	急性黄疸型肝炎	温和灸	太冲
14	震颤麻痹	隔姜灸	膈俞
15	震颤麻痹	隔姜灸	肝俞
16	震颤麻痹	隔姜灸	脾俞
17	震颤麻痹	隔姜灸	命门
18	震颤麻痹	雷火针灸	气海俞
19	震颤麻痹	雷火针灸	足三里
20	慢性毒性肝炎	温和灸	阳陵泉
21	慢性毒性肝炎	温和灸	肝俞

此后，我们使用了Neo4j图数据库进行数据存储。 Neo4j是一个高性能的，NOSQL图形数据库，它将结构化数据存储在网络上而不是表中。它是一个嵌入式的、基于磁盘的、具备完全的事务特性的Java持久化引擎，但是它将结构化数据存储在图上而不是表中 。（例如下方是Neo4j数据库的可视化页面）



针对三元关系，团队成员讨论决定抽取出其中的五组二元关系建立节点连接，分别为

节点1类型	关系（单向）	节点2类型
病症	使用	灸法
灸法	治疗	病症
灸法	针对	穴位
病症	对应穴位	穴位
穴位	关联	病症

同时，针对各个节点与各自类别的联系，我们为其同样建立了联系

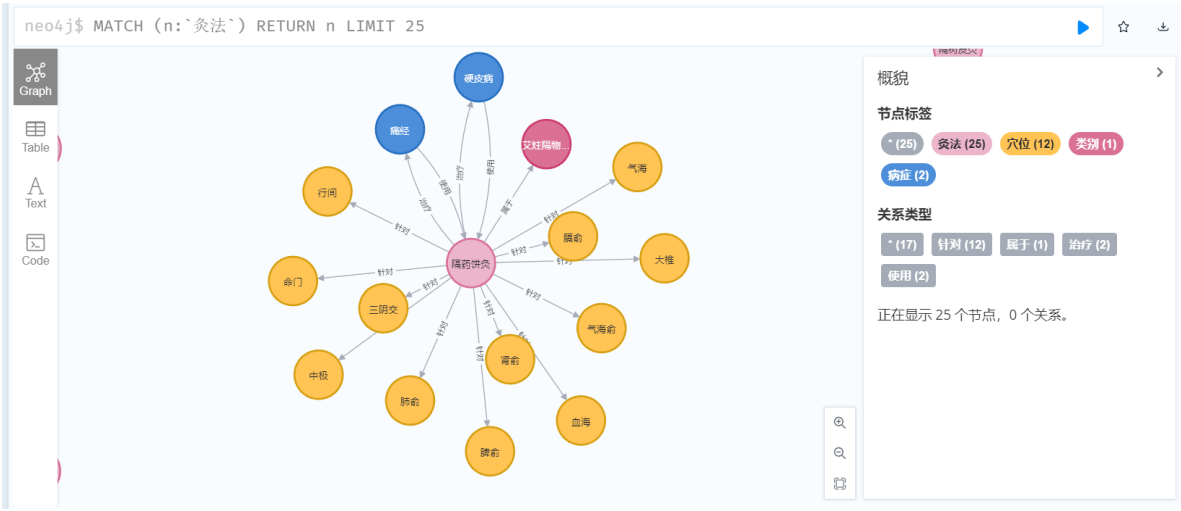
节点1类型	关系（单向）	节点2类型
灸法/病症/穴位	属于	类别

在整理出以上六组单向边后，团队成员使用Python的py2neo库作为辅助，将上述的所有整理资料导入到Neo4j数据库中，部分对应代码如下

```
# 获取上文提及的excel文件，0列为病症名称，1列为灸法名称，2列为病症名称
# graph: Neo4j图数据库
relation_file_path = 'static/datas/关联表.xls'
def create_disease_jiufa_xuewei_graph():
    data = pd.read_excel(relation_file_path, sheet_name="Sheet1", header=0)
    data_array = np.array(disease_jiufa_xuewei_data)
    # 通过name为索引寻找节点
    for i in range(0, len(disease_jiufa_xuewei_data_array)):
        bingzheng_node = graph.nodes.match(name=data_array[i][0]).first()
```

```
        jiufa_node = graph.nodes.match(name=data_array[i][1]).first()
        xuewei_node = graph.nodes.match(name=data_array[i][2]).first()
        # 建立病症灸法间关系Relationship
        relation = Relationship(bingzheng_node, "使用", jiufa_node)
        graph.create(relation)
        relation = Relationship(jiufa_node, "治疗", bingzheng_node)
        graph.create(relation)
        # 建立灸法穴位间关系Relationship
        relation = Relationship(jiufa_node, "针对", xuewei_node)
        graph.create(relation)
        # 建立病症穴位间关系Relationship
        relation = Relationship(bingzheng_node, "对应穴位", xuewei_node)
        graph.create(relation)
        relation = Relationship(xuewei_node, "关联", bingzheng_node)
        graph.create(relation)
```

在生成之后，Neo4j中的可视化图案如下展示：



在形成Neo4j对应的数据库后，团队成员便可通过Neo4j的查询语句进行查询，来针对返回结果进行展示。

4.3 数据库介绍

Neo4j中的数据格式以及介绍如下：

节点：

节点类型	节点介绍	内置属性	内置属性
灸法	用于存储每一条灸法的信息，例如：（19，隔药饼灸）	id：number	name：string
病症	用于存储每一条病症的信息，例如：（8，流行性出血热）	id：number	name：string
穴位	用于存储每一条穴位的信息，例如：（192，头临泣）	id：number	name：string
类别	用于存储每一种种类的信息，例如：（足太阴脾经穴）	/	name：string

关系：

关系名称	关系介绍	始节点类型	终节点类型
使用	病症应使用哪些灸法	病症	灸法
治疗	灸法能治疗哪些病症	灸法	病症
关联	灸法能关联到哪些穴位	灸法	穴位
对应穴位	病症治疗对应哪些穴位	病症	穴位
针对	穴位能够针对那些病症	穴位	病症

4.4 数据展示

> 对应文件：前端/src/GraphView/GraphInfor.vue

本团队项目的数据展示主要采用Vue + echarts进行。

首先，用户输入对应搜索名称/选择节点/切换节点时，均会触发searchWord函数，该函数接收name，mode两个参数，前者代表查找节点的名称，后者代表查找节点的类型（病症、灸法或者穴位），该函数会针对传入的name和mode选择不同的neo4j查询语句来进行处理：

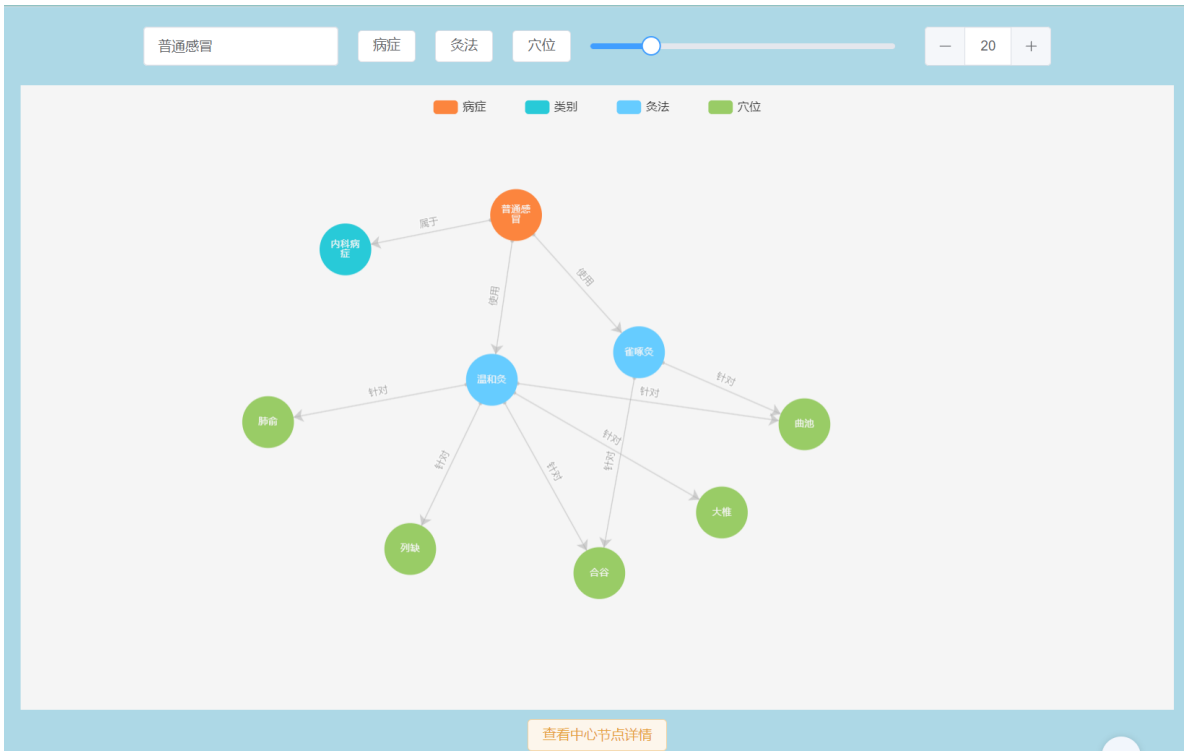
searchWord函数：

```
async searchWord(name, mode){
    var word = name
    this.searchInput.word = name
    this.searchInput.mode = mode
    var limit = this.searchInput.limit
    var query = ""
    if(mode === "病症"){
        query = "MATCH p=(d:病症{name:'"+word+"'})-[:`使用`]->(j:灸法)-[:`针对`]->(x:穴位) WHERE (x)-[:关联]->(d) MERGE q=(d)-[:`属于`]->(l:`类别`) RETURN q,p LIMIT " + limit
    }
    else if(mode === "灸法"){
        query = "MATCH p=(j:灸法{name:'"+word+"'})-[:`针对`]->(x:穴位)-[:`关联`]->(b:`病症`) WHERE (b)-[:`使用`]->(j) MERGE q=(j)-[:`属于`]->(l:`类别`) RETURN q,p LIMIT " + limit
    }
    else{
        query = "MATCH p=(x:穴位{name:'"+word+"'})-[:`关联`]->(b:`病症`)-[:`使用`]->(j:`灸法`) WHERE (j)-[:`针对`]->(x) MERGE q=(x)-[:`属于`]->(l:`类别`) RETURN q,p LIMIT " + limit
    }
    this.executeCypher(query);
},
```

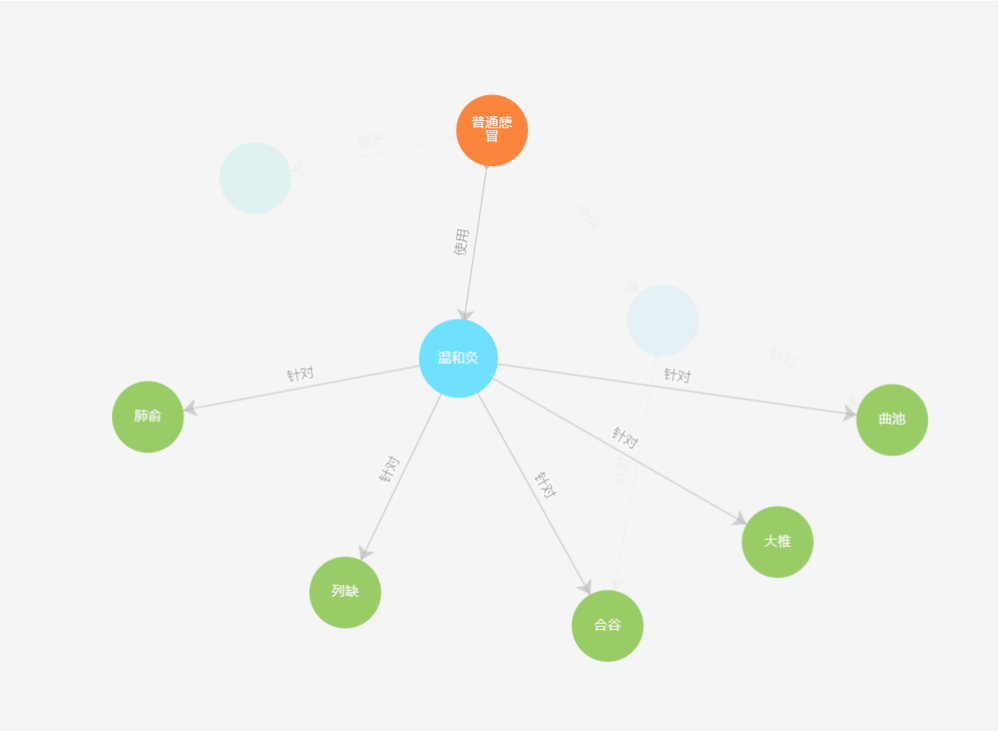
> 在这里对语句进行简单介绍：
>
> ```
> query = "MATCH p=(d:病症{name:'"+word+"'})-[:`使用`]->(j:灸法)-[:`针对`]->(x:穴位) WHERE (x)-[:关联]->(d) RETURN p"
> ```
>
> 该语句是病症的查询语句，系统接受一个word并将其拼接到查询语句中，执行该语句时，neo4j数据库会去病症节点中寻找name为word的节点，并且去关联到该节点所**使用**的所有**灸法**，再通过灸法去关联到该灸法所**针对**的所有**穴位**，并且，对后者关系进行一个约束，即这些穴位要同时能**关联**到word的病症节点。

> 因此，该语句将返回：**治疗当前病症可以使用的所有灸法，以及每个灸法针对该病症使用的穴位**

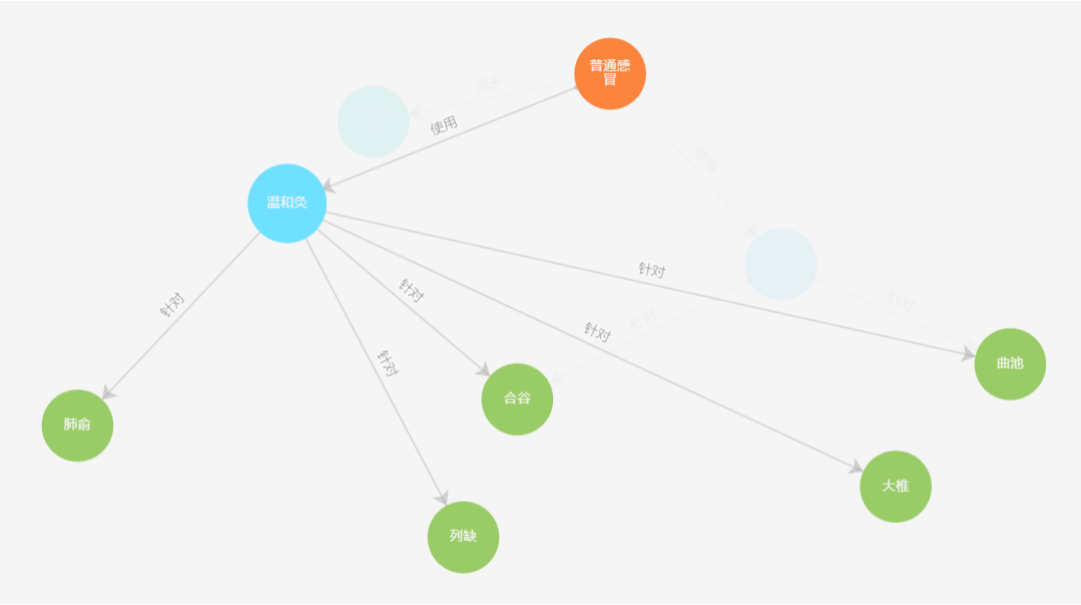
executeCypher函数则会将该查询语句发送给Neo4j，并对返回结果进行分析和组装，将其整理成Echarts可以接受的数据格式，Echarts将对所有返回边进行渲染，并将其回显到页面，例如下方是普通感冒的回显结果：



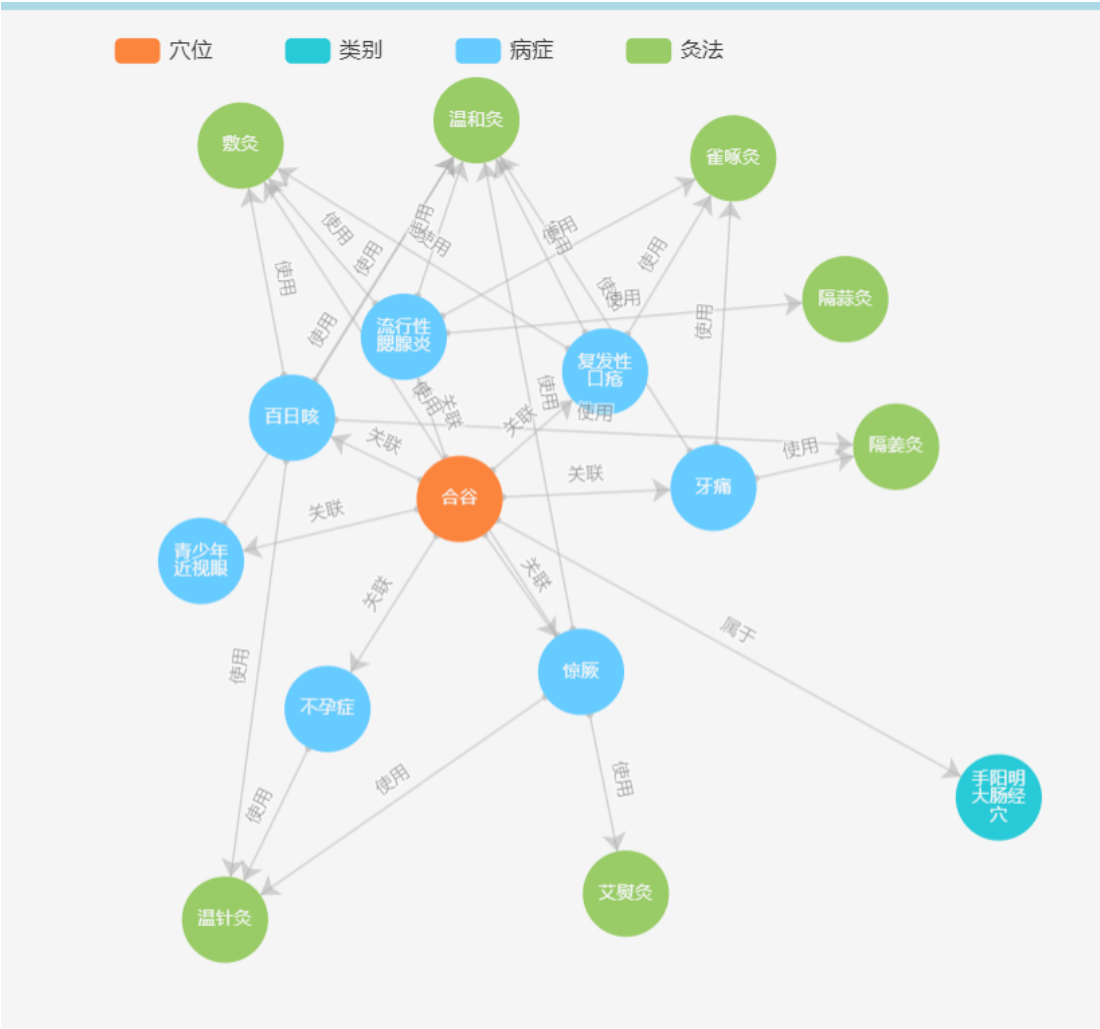
同时，团队成员利用了Echarts的内置功能，允许用户将鼠标移动到节点上方并查看当前节点的关联边（无关边会被隐藏）：（例如下方，鼠标移动到温和灸上方，可以看到其它雀啄灸和内科病症的信息被隐藏）



用户同时可以针对节点进行拖拽：（例如下方，拖动了温和灸）

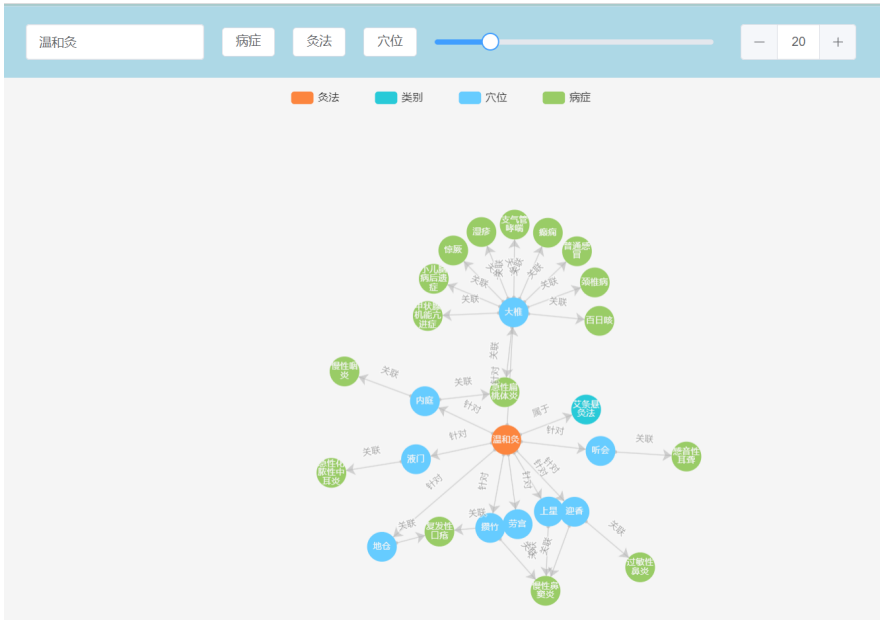


如果双击节点，则会以当前双击节点为中心重新调用searchWord进行搜索：（例如下方，双击了合谷节点，页面刷新为合谷为中心的搜索结果）

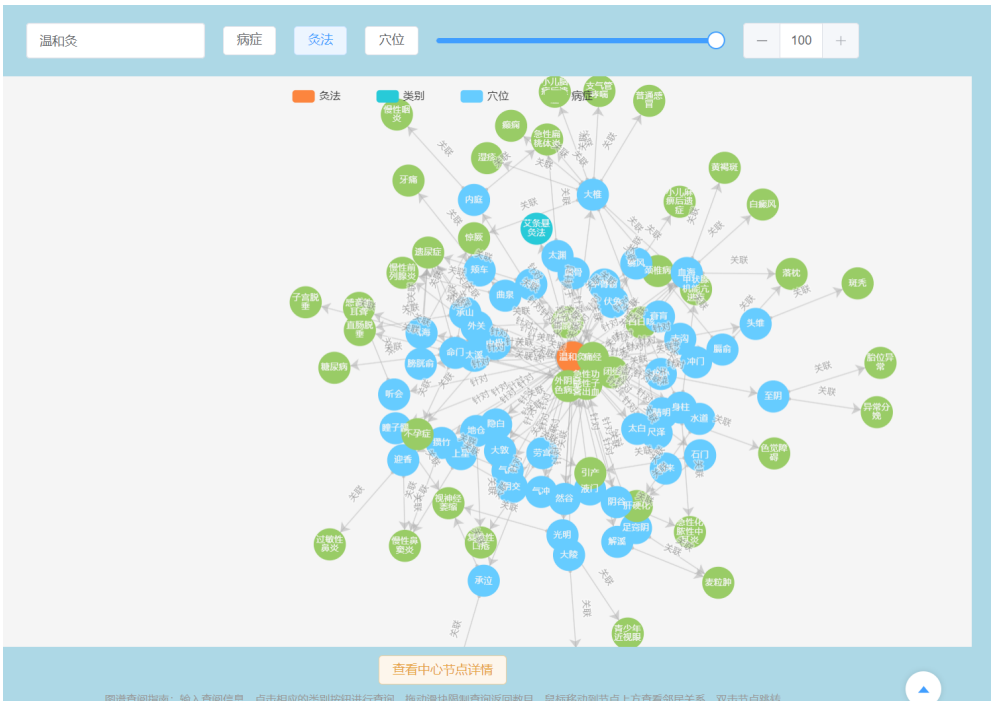


同时，团队成员制作了滑块来方便用户调整回显数量：（下方分别展示了温和灸显示20/100条的结果）

20条：



100条：



如果图谱过大/过小，用户可以通过滚动鼠标滚轮来调整图的展示大小。

5.用户中心