第十七章 DNA 搜索

第一节 DNA 搜索的动机

一开始我的 DNA 搜索动机是能将元基肽展公式进行真实生产环境实践，体现生产力结构的升级和改变，慢慢的我的动机开始模糊，探索一种思维量化模式的搜索引擎，提高计算理解力，

第二节 DNA 搜索的应用需求

我的需求很简单，就是满足养疗经[17]的一切应用的基础提供保障，这个保障我归纳为数据保障，算法保障，应用保障.

1 数据保障，医学的数据需要严谨的，有序的，完整的知识结构. 避免搜索断层，出现分歧

2 算法保障，每一个算法，要有对应功能的靶性，和计算力可持续优化的结构（微分催化[7]结构）

3 应用保障，主要是医学实践 和 医学应用的傻瓜化，体现在快速理解和迅捷方便的准确（快广准） 应用.

第三节 DNA 搜索的具体描述

```
public void zhongYaoSearch(String zhongyao, String forE, String key)
        { String CatalyticDNA_xingwei = "";
        String CatalyticDNA_gongxiao= "";
        if(DNASearchIsClick) {
            TokenPDI pDE_RNA_Formular= new TokenPDI();
            double rate= ((double)催化比值rot)/ 100;
            //pDE_RNA_Formular.key[0]= 0.3;
            pDE_RNA_Formular.key[1]= rate;
            pDE_RNA_Formular.key[2]= rate;
            //pDE_RNA_Formular.key[3]= 0.3;
            if(null!= name_feel_filter.getText()) {
                pDE_RNA_Formular.pdw= name_feel_filter.getText().toUpperCase();
                pDE_RNA_Formular.code= pDE_RNA_Formular.pdw.toString().toUpperCase();
                pDE_RNA_Formular.doKeyPress(pDE_RNA_Formular.code, pDE_RNA_Formular, false);

                CatalyticDNA_xingwei= null== pDE_RNA_Formular.pde?""
                    : pDE_RNA_Formular.pde.toString();
                }
            //
```

```
        pDE_RNA_Formular.pde= "";
        if(null!= name_filter_not_have.getText()) {
            pDE_RNA_Formular.pdw= name_filter_not_have.getText().toUpperCase();
            pDE_RNA_Formular.code= pDE_RNA_Formular.pdw.toString().toUpperCase();
            pDE_RNA_Formular.doKeyPress(pDE_RNA_Formular.code, pDE_RNA_Formular, false);

            CatalyticDNA_gongxiao= null== pDE_RNA_Formular.pde?""
                : pDE_RNA_Formular.pde.toString();
        }
    }
    String[] score= new String[copy.size()];
    int[] score_code= new int[copy.size()];
    double []reg= new double[copy.size()];
    int count= 0;
    Map<String, WordFrequency> mapSearchWithoutSort= null;
    if(dic_map.containsKey(zhongyao.replaceAll(" ", ""))) {
        mapSearchWithoutSort= analyzer.parserMixStringByReturnFrequencyMap(zhongyao);
    }else {
        if(key.split(" ")[0].length()> 5) {
            mapSearchWithoutSort= analyzer.parserMixStringByReturnFrequencyMap(key);
        }else {
            mapSearchWithoutSort= analyzer.parserMixStringByReturnFrequencyMap(zhongyao);

        }
    }
    Iterator<String> iteratorForCopy= copy.iterator();
    int copyCount= 0;
    List<String> list= analyzer.parserMixedString(key);
    String[] string= ListSwap.listToArray(list);

    String[] stringReg= new String[forE.length()/ 3];
    for(int i= 0; i< stringReg.length; i++) {
        stringReg[i]= forE.substring(i* 3, (i* 3+ 3)< forE.length()
            ? (i* 3+ 3): forE.length()- 1);
    }
    while(iteratorForCopy.hasNext()) {
        String iteratorForCopyString= iteratorForCopy.next();
        score[copyCount]= iteratorForCopyString;
        String temps=  dic_map.get(iteratorForCopyString).toString();
        String tempsPCA= dic_li.get(iteratorForCopyString).toString();
        String tempsIndex= dic_index.get(iteratorForCopyString).toString();
        Iterator<String> iteratorWordFrequency= mapSearchWithoutSort.keySet().iterator();
        Here:
            while(iteratorWordFrequency.hasNext()) {
                String mapSearchaAtII = iteratorWordFrequency.next();
```

```
                WordFrequency wordFrequencySearch = mapSearchWithoutSort.get(mapSearchaAtII);
                if(temps.contains(mapSearchaAtII)) {
                    if(reg[copyCount] == 0){
                        count += 1;
                    }
                    if(score[copyCount].contains(zhongyao.replace(" ", "")))
                        { reg[copyCount]+= 12;
                    }
                    if(zhongyao.contains(score[copyCount].replace(" ", "")))
                        { reg[copyCount]+= 12;
                    }

                    if(tempsIndex.equalsIgnoreCase(zhongyao.replace(" ", "")))
                        { reg[copyCount]+= 1200;
                    }
                    if(tempsIndex.contains(zhongyao.replace(" ", "").toUpperCase()))
                        { reg[copyCount]+= 1200;
                    }

                    score[copyCount]= iteratorForCopyString;
                    if(!pos.containsKey(mapSearchaAtII))
                        { reg[copyCount]+= 1;
                        score_code[copyCount]+= 1<< mapSearchaAtII.length()
                                << wordFrequencySearch.getFrequency() ;
                        if(tempsPCA.contains(mapSearchaAtII))
                            { score_code[copyCount] *= 2;
                        }
                        if(score[copyCount].contains(mapSearchaAtII)) {
                            if(score[copyCount].length()>1)
                                { reg[copyCount]+= 22;
                            }
                            reg[copyCount]+= 3;
                        }
                        continue Here;
                    }
                    if(pos.get(mapSearchaAtII).contains("名")
                            || pos.get(mapSearchaAtII).contains("动")
                            || pos.get(mapSearchaAtII).contains("形")
                            || pos.get(mapSearchaAtII).contains("谓"))
                        { reg[copyCount]+= 2;
                        if(tempsPCA.contains(mapSearchaAtII)) {
                            reg[copyCount]*= 2;
                        }
                    }
                    reg[copyCount]+= 1;
```

```
                        score_code[copyCount]+= (iteratorForCopyString.contains(mapSearchaAtII)?
2: 1)
                                * (!pos.get(mapSearchaAtII).contains("名")
                                ? pos.get(mapSearchaAtII).contains("动")? 10: 1: 150)
                                << mapSearchaAtII.length()* wordFrequencySearch.getFrequency();
                        if(score[copyCount].contains(mapSearchaAtII)) {
                            if(score[copyCount].length()>1)
                                { reg[copyCount]+= 22;
                            }
                            reg[copyCount]+= 3;
                        }
                        continue Here;
                    }
                    if(mapSearchaAtII.length()>1) {
                        for(int j=0; j<mapSearchaAtII.length(); j++) {
                            if(temps.contains(String.valueOf(mapSearchaAtII.charAt(j)))) {
                                if(reg[copyCount] == 0){
                                    count += 1;
                                }
                                score[copyCount] = iteratorForCopyString;
                                score_code[copyCount]+=1;
                                if(pos.containsKey(String.valueOf(mapSearchaAtII.charAt(j)))
&&(pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("名")
        ||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("动")
        ||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("形")
                    ||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("谓")
                                    )) {
                                    reg[copyCount] += 2;
                                }
                                reg[copyCount] += 1;
                                if(score[copyCount].contains(mapSearchaAtII)) {
                                    if(score[copyCount].length()>1)
                                        { reg[copyCount]+= 12;
                                    }
                                    reg[copyCount]+= 3;
                                }
                                continue Here;
                            }
                        }
                    }
                }
            score_code[copyCount] = score_code[copyCount] * (int)reg[copyCount];
            //词距
            int code= 200;
            int tempb= 0;
```

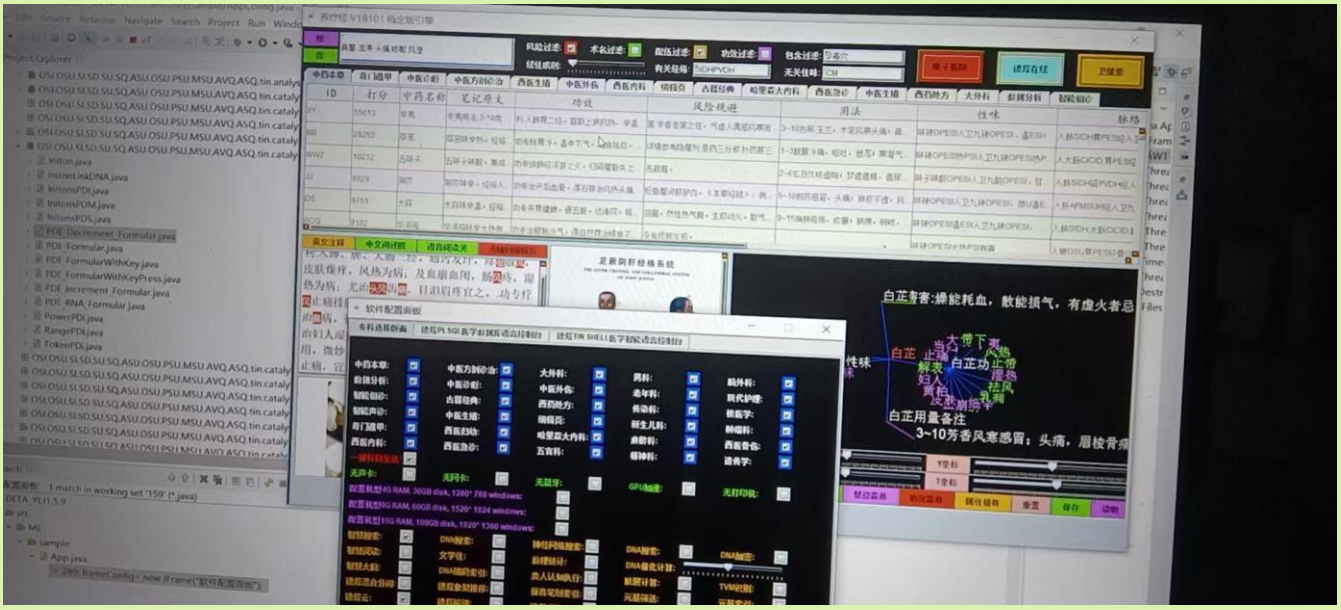
```
        int tempa= score_code[copyCount];
        if(key.length()> 6) {
            //全词
            for(int i= 0; i< string.length; i++) {
                if(temps.contains(string[i]))
                    { tempb+= code;
                }
            }
            //断句
            for(int i= 0; i< stringReg.length; i++) {
                if(temps.contains(stringReg[i]))
                    { tempb+= code;
                }
            }
            score_code[copyCount] = (int) (tempa/Math.pow(lookrot+ 1, 4)
                 + tempb*Math.pow(lookrot, 2));
        }
        if(zhongyao.replace(" ", "").length()> 1&& zhongyao.replace(" ", "")
            .length()< 5) {
            if(temps.contains(zhongyao.replace(" ", "")))
                { tempb+= code<< 7;
            }
            score_code[copyCount] = (int) (tempa/Math.pow(lookrot+ 1, 4)
                + tempb*Math.pow(lookrot, 2));
        }
        copyCount++;
    }
    new Quick9DLYGWithStringSwap().sort(score_code, score);
    Object[][] tableData= new Object[count][13];
    int new_count = 0;
    newTableModel.getDataVector().clear();
    if(null== key|| key.equals("")) {
        for(int i= 0; i < tableData_old.length; i++)
            { tableData_old[i][6]= tableData_old[i][6]==null?
            ""
            : tableData_old[i][6];
            newTableModel.insertRow(i, tableData_old[i]);
        }
        newTableModel.fireTableDataChanged();
        return;
    }
    Here:
        for(int i = copy.size()-1; i > -1; i--) {
            if(score_code[i]< 1){
                continue Here;
            }
```

```
            if(risk_filter_box.isSelected()) {
                String hai= (dic_hai.get(score[i])==null?"null."
                        : dic_hai.get(score[i]).toString().replaceAll("\\s*", "")
                        .equalsIgnoreCase("")?"null": dic_hai.get(score[i]).toString()
                                .replaceAll("\\s*", ""));
                String temp= name_filter.getText();
                for(int j=0; j<temp.length(); j++) {
                    if(hai.contains(""+ temp.charAt(j))) {
                        continue Here;
                    }
                }
            }
            if(feel_filter_box.isSelected()) {
                String li= (dic_li.get(score[i])==null?"null."
                        : dic_li.get(score[i]).toString().replaceAll("\\s*", "")
                        .equalsIgnoreCase("")?"null": dic_li.get(score[i]).toString()
                                .replaceAll("\\s*", ""));
                String temp= name_filter.getText();
                for(int j= 0; j< temp.length(); j++) {
                    if(li.contains(""+ temp.charAt(j))) {
                        continue Here;
                    }
                }
            }
            //催化比值rot dna 催化计算
            if(!name_feel_filter.getText().isEmpty()) {
                String wei= dic_xw.get(score[i]).toString().replaceAll("\\s*", "");
                CatalyticDNA_xingwei+= name_feel_filter.getText().replace(" ", "");
                for(int j= 0; j< CatalyticDNA_xingwei.length(); j++) {
                    if(wei.contains(""+ CatalyticDNA_xingwei.charAt(j))) {
                        continue Here;
                    }
                }
            }
            if(null!= name_filter_not_have.getText()) {
                if(!name_filter_not_have.getText().replace(" ", "").isEmpty())
                    { String wei= dic_jm.get(score[i]).toString().replaceAll("\\s*",
                    "");
                    CatalyticDNA_gongxiao+= name_filter_not_have.getText().replace(" ", ""); ;
                    for(int j= 0; j< CatalyticDNA_gongxiao.length(); j++) {
                        if(!wei.contains(""+ CatalyticDNA_gongxiao.charAt(j))) {
                            continue Here;
                        }
                    }
                }
            }
```

```
            if(shuming_filter_box.isSelected())
                { String wei= score[i];
                String temp= name_filter.getText();
                for(int j= 0; j< temp.length(); j++) {
                    if(wei.contains(""+ temp.charAt(j))) {
                        continue Here;
                    }
                }
            }
            String temp= dic_map.get(score[i]).toString();
            if(tableData.length<= new_count) {
                continue Here;
            }
            tableData[new_count]= new
                    Object[]{ (dic_index.get(score[i])=
                    = null? ""
                    : dic_index.get(score[i])).toString().replaceAll("\\s*", ""),
                    score_code[i], score[i],
                    (dic_yw.get(score[i])== null? ""
                    : dic_yw.get(score[i])).toString().replaceAll("\\s*", ""),
                    (dic_li.get(score[i])== null? ""
                    : dic_li.get(score[i])).toString().replaceAll("\\s*", ""),
                    (dic_hai.get(score[i])==null?
                    "详情参考笔记原文列：是药三分毒，补药甚三分．食材亦如此，勤俭亦长生."
                    : dic_hai.get(score[i]).toString().replaceAll(\\s*
                    , "").equalsIgnoreCase("")?"详情参考笔记原文列"
                    : dic_hai.get(score[i]).toString().replaceAll("\\s*", "")),
                    (dic_yl.get(score[i])==null?"详情参考相关书籍"
                        : dic_yl.get(score[i])).toString().replaceAll("\\s*", ""),
                    (dic_xw.get(score[i])== null? ""
                        : dic_xw.get(score[i])).toString().replaceAll("\\s*", ""),
                    (dic_jm.get(score[i])== null? ""
                        : dic_jm.get(score[i])).toString().replaceAll("\\s*", ""),
(dic_xz.get(score[i])==null?"": dic_xz.get(score[i])).toString().replaceAll("\\s*", "")
, (dic_jj.get(score[i])==null?"": dic_jj.get(score[i])).toString().replaceAll("\\s*", "")
, (dic_cy.get(score[i])==null?"": dic_cy.get(score[i])).toString().replaceAll("\\s*", "")
, (dic_ya.get(score[i])==null?"": dic_ya.get(score[i])).toString().replaceAll("\\s*", "")
, (dic_zf.get(score[i])==null?"": dic_zf.get(score[i])).toString().replaceAll("\\s*", "")
, (dic_cj.get(score[i])==null?"": dic_cj.get(score[i])).toString().replaceAll("\\s*", "")};
            if(zhongyao.contains("风寒")) {
                if(temp.contains(" 风 寒 "))
                    { newTableModel.insertRow(new_count,
                    tableData[new_count]); new_count += 1;
                }
            }else if(zhongyao.contains("风热
                ")){ if(temp.contains("风热")) {
```

```
                    newTableModel.insertRow(new_count, tableData[new_count]);
                    new_count += 1;
                }
            }else {
                newTableModel.insertRow(new_count, tableData[new_count]);
                new_count+=1;
            }
        }
    newTableModel.fireTableDataChanged();
                    }
```

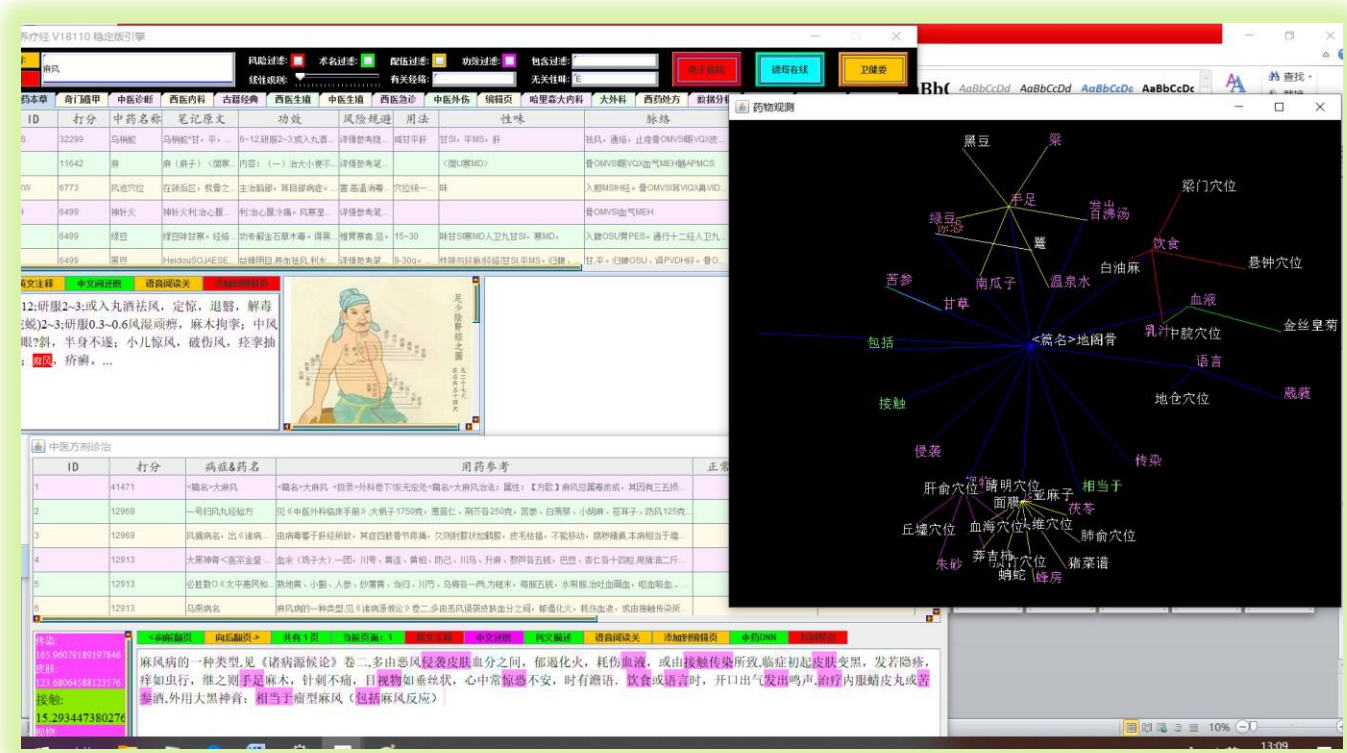
第四节 DNA 搜索的应用实现



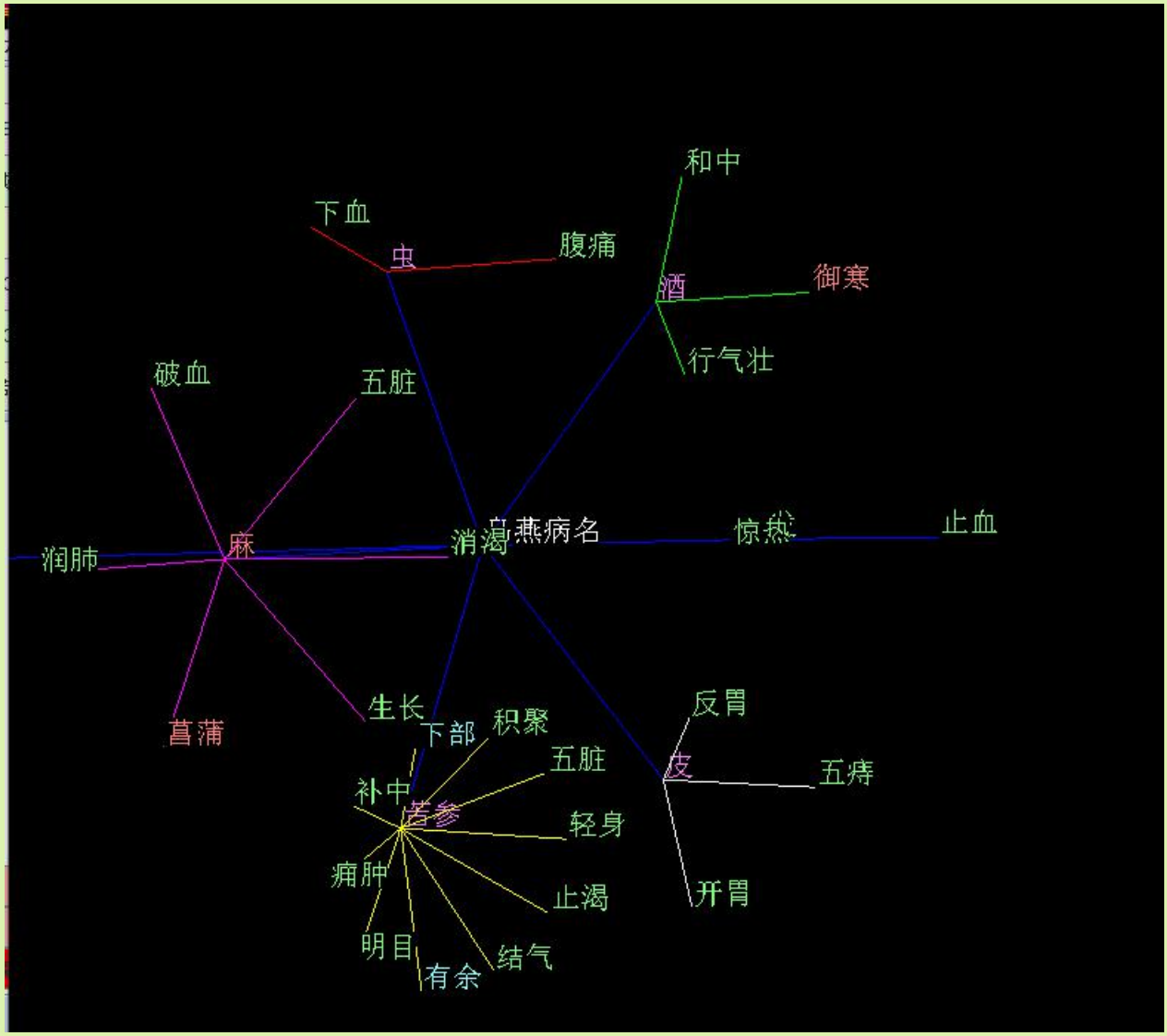
在没有设计元基之前, 我便设计了多种完整的数据聚类搜索函数. 我一直在思考, （稍后我会加非元基搜索的多种实例展示. ）如果加入了元基计算, 要达到怎么样一种预期效果, 快, 广, 准, 是必要的.其次？ 我想到很多, 但终究不如实践与推导. 于是我设计了这个计算模式的版本如图. 效果不错, 但很粗糙, 因为是我的第一代元基搜索. 我一会一直优化它.

这个两张搜索 关于风湿性风寒, 筛选出了白芷, 我结合中医看了下, 筛出的都是君药, 我之后会按书上的意思推选出臣药出来. 让计算观测功能更加丰富, 严谨, 准确.

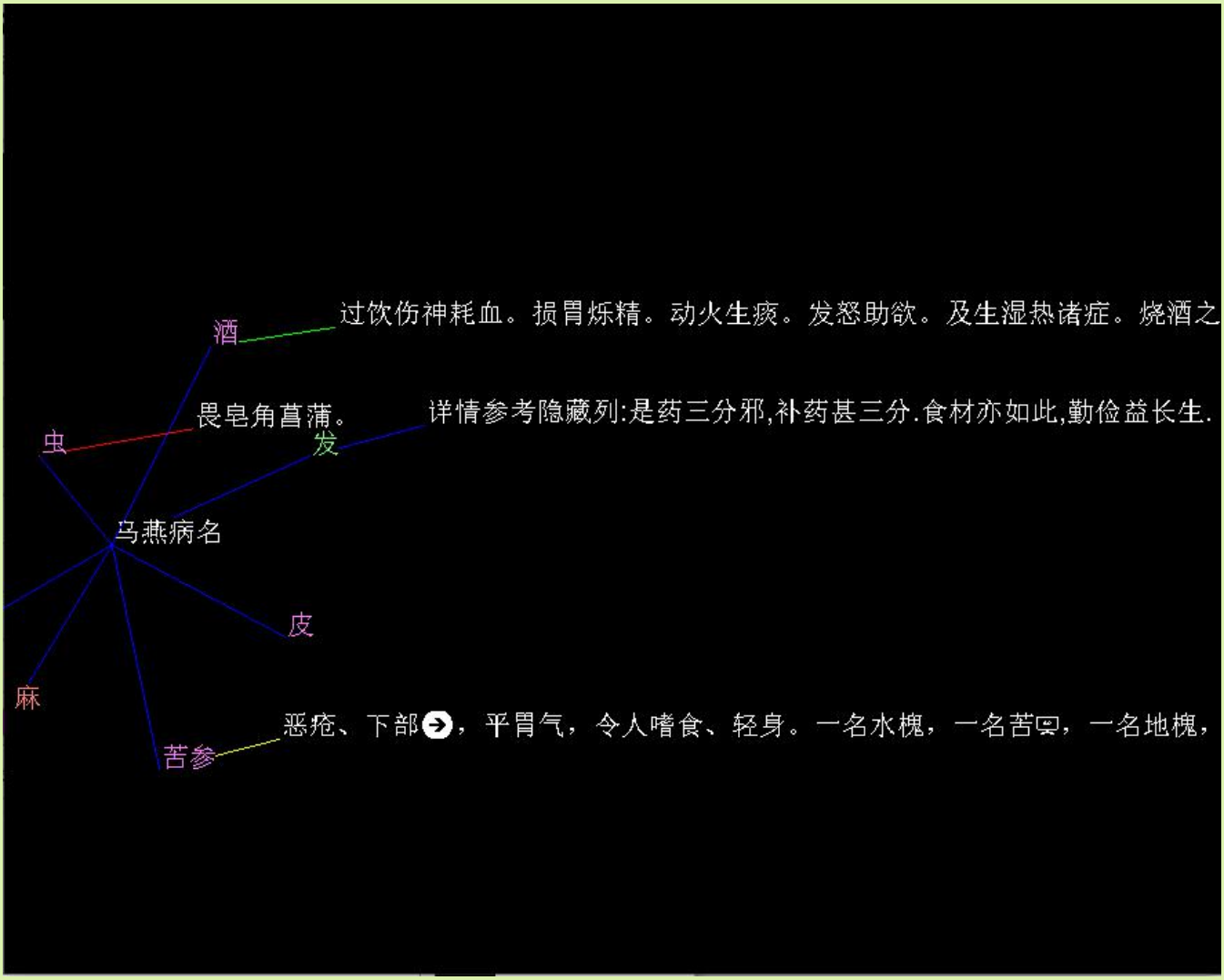
简单介绍下中药本草页面的逻辑上面横条是全局栏目, 用于 搜索, 筛选, 观测和全局控制, 中药页的上部分是一个六行显示的表格, 默认排序输出价值元组. 下面则是数据观测部分, 从一维的线性文本, 2 维的图片数据, 到三维的属性花, 组成了数据分析的核心部分. 现在, 经络和性味的元基筛选已经成功, 如上图展示.



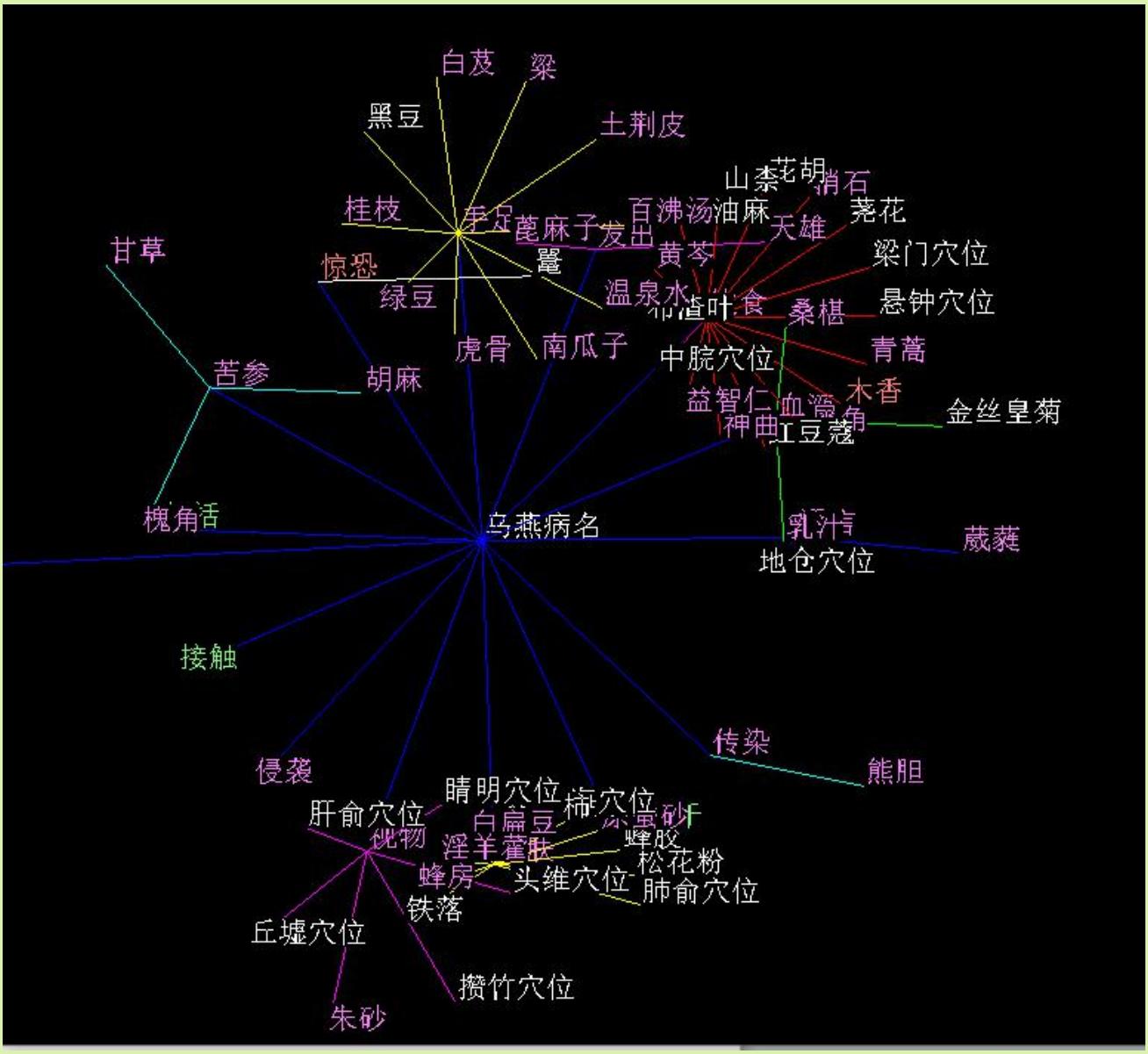
上图通过无关性味的E 元基过滤后, 搜索麻风 列出的中医方剂进行 DNN 深度分析价值词汇, 然后进行三维属性聚类观测, 此时, 中医药页面显示其相关对症的有价值中药列次在表格中方便查阅. 之后这元基变换函数我会升级在数据变换 API[3]中。同时，上面的应用是中医方剂和中医本草两个页面进行耦合操作，然后在三维图中观测，养疗经的全局分类依次打分排序体系支持多种复杂的分析操作。



这张图的原理是点击方剂元组后, DNN 展示的数据词汇（根据不同的精度, 词汇数目可以控制） 进行分类, 分类后的第一层展开, 开始进行第二层功效搜索, 进行聚类. 这样一分一聚 就生成了DNN 三维词汇花的骨架结构. 如图 乌燕一药含有 苦参, 苦参有治疗痈肿的功效.



这张图方便在药物聚类搜索后查看其主要禁忌, 于是我设计了这个功能组件. 之后禁忌属性也会全部语义方式元基肽化.



这张图恰好与 DNN 观测相反. 我将乌燕一方剂进行主要功效拓扑, 然后每一个有价值功效分类进行 相关对症的药物聚类展示, 如图, 乌燕对传染性疾病有价值, 传染病对症的药物有熊胆（黄疸类肺皮血肝胆部等疾病）. 入手足疾病, 桂花治疗手痛. 如果关联有疑惑, 于是可以通过中药页表格搜索进行持续傻瓜化搜索.