空间数据的即敲即得式检索

2011011258　　计 13　谭志鹏

实验功能实现

1、 基于位置的检索，找出与输入前缀匹配的距离中心点最近的若干城市结果，能够实时显示结果。通过点击可以改变地图中心点。
2、 即敲即得式检索，输入过程实时显示查询结果，给出搜索建议。
3、 支持多单词分别进行前缀匹配。
4、 良好的界面效果与交互效果。

实验框架描述

1、前端

1）调用 google API 显示地图，能够支持鼠标的拖动以及放大缩小的效果。

2）使用 javascript+html+css 实现网页端的显示以及处理用户行为。其中使用了 jquery 作为框架。在界面显示效果上做了较多优化。

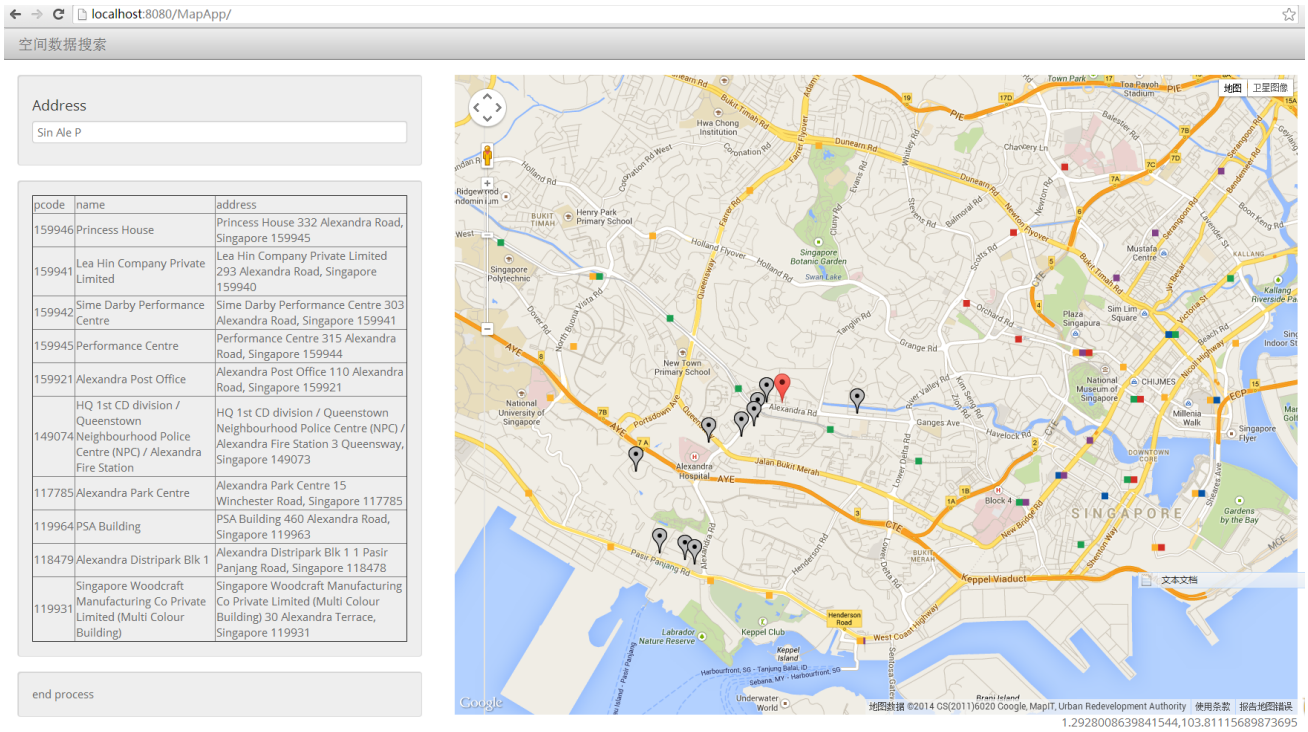
2、后端

1）使用 java 语言继承 HttpServlet 作为服务器端。通过 doPost 方法处理前端的查询请求。

2）使用 Tomcat 搭建服务器平台。

3）算法上采用前缀 Trie 树筛选出符合前缀要求的结果，再利用 kdtree 来实现空间数据的快速搜索。

效果展示：



其中左上角进行输入，结果会在右侧地图上实时显示，同时输入框下面还会以表格形式显示出距离中心点最近的十个查询结果的详细信息。鼠标移到每个标记的位置上就能看到该地点的名字。

实验算法设计

后台的搜索算法框架如下：

1、 对于实验数据进行预处理，建立基于前缀索引的 Trie 树。

关键程序如下：

```
public void buildupData(){
    for(int i=0; i<dataVec.size(); i++){
        JSONObject data = dataVec.get(i);
        String dataStr = data.getString("addr") + " " + data.getString("name");
        String grams[] = dataStr.split("[\\s]");

        for(int j=0; j<grams.length; j++){
            prefixTrie.addGram(grams[j], i);
        }
    }
}
```

2、 对于每一个查询串，分拆得到多个前缀 gram。对于每个 gram，在 Trie 树中基于前缀查找到包含这些 gram 的单词的索引号，获得所有满足前缀条件的候选地点。

3、 对这些候选及诶单建立 kdtree，利用 kdtree 能够快速找出离中心点最近的若干个结果。其中 kdtree 使用了网上的开源程序。

关键程序如下：

```
public List<Entry<JSONObject>> searchQuery(String query){
    String [] frefixGrams = query.split("[\\s]");
    Set<Integer> candidates = null;

    //search candidates using prefixTrie
    for(int i=0; i<frefixGrams.length; i++){
        Set<Integer> tCandidates  = prefixTrie.prefixSearch(frefixGrams[i]);
        if(tCandidates == null) return null;
        if(candidates == null)
            candidates = tCandidates;
        else
            candidates.retainAll(tCandidates);
    }
    //build kdtree for the candidates
    SqrEuclid<JSONObject> tKdTree = new SqrEuclid<>(2, candidates.size()+100);
    for (Integer index : candidates){
        JSONObject jsonCandi = dataVec.get(index);
        JSONArray jsonLoc = (JSONArray)jsonCandi.get("latlng");
        double[] location = new double[2];
        location[0] = jsonLoc.getDouble(0);
        location[1] = jsonLoc.getDouble(1);
        tKdTree.addPoint(location, jsonCandi);
    }

    //find nearest points using the kdtree
    List<Entry<JSONObject>> result = tKdTree.nearestNeighbor(centerLoc, 10, true);
    Collections.reverse(result);
    return result;
}
```

实验效果

基于上述算法下实现的应用在搜索过程中能够很好的做到实时计算输出，不会感觉到明显的停顿，能够给用户比较好的使用体验。说明上述算法处理 100w 数据能够达到实时处理的要求。

实验总结

通过这次实验，最大的难点应该在于框架的搭建，相应的后台算法倒是更简单一些。由于自己之前没有写过完整的 web 应用，所以自己从无到有学习了如何编写 javascript, 如何搭建 web 服务器等知识。这些知识虽然不是很难，但是相应的技巧性很巧，作业过程中查阅了很多相关资料，借鉴了很多网上的一些模板和例子。通过这次实验对于 web 应用的前端后端的编写有了比较多的了解，这也算一大收获。

另外通过这次实验对于空间数据的搜索方法以及即敲即得式的搜索有了一次实践的机会。参照课件上的讲解还是比较容易理解的。

可以说通过一个学期的学习，自己对于数据库领域的前沿课题有了比较深入的了解，而且通过小作业大作业得到了具体的实践机会，收获很大。最后感谢老师一个学期的辛勤指导。还要感谢助教一直以来的耐心帮助，每次遇到问题发邮件都能够很及时的给予解答。