1. **实验目的和要求**
2. 爬取京东上的1000条评论
3. 将爬取的信息格式化
4. **实验任务**

爬取京东或淘宝某一商品的评论1000条，使用jieba分词等分词工具分词，统计词频（使用MapReduce或HBase或Hive），并以词云、统计表的方式可视化呈现。

1. **实验准备方案**

包括以下内容：（硬件类实验：实验原理、实验线路、设计方案等）

（软件类实验：所采用的系统、组件、工具、核心方法、框架或流程图、程序清单等）

实验采用的系统：centos 6.5、hadoop2.5.2

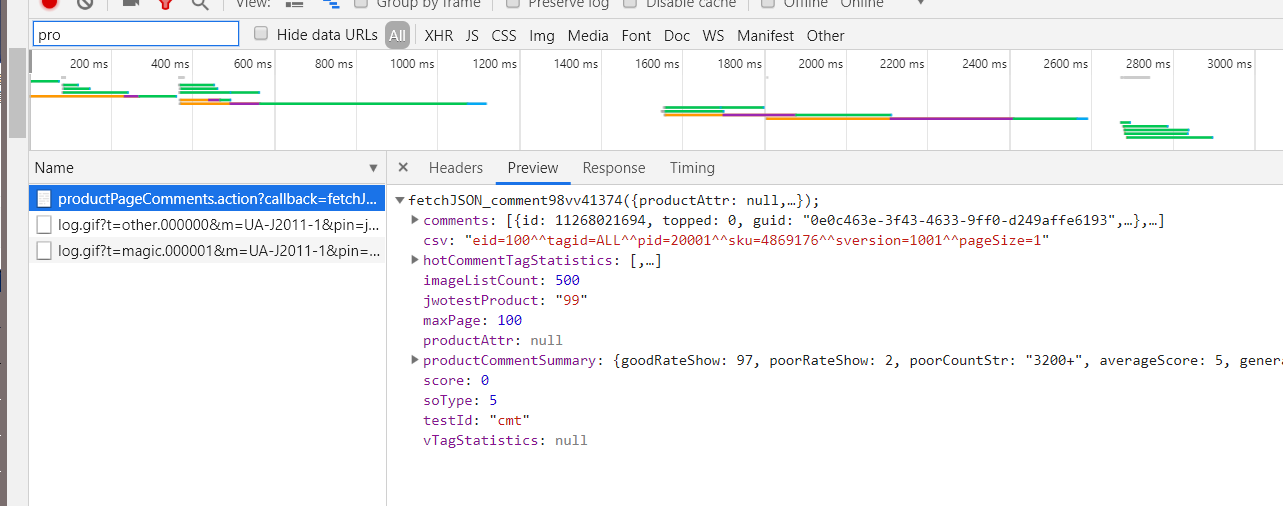
工具：VMware WorkStation

1. **实验用仪器、设备**

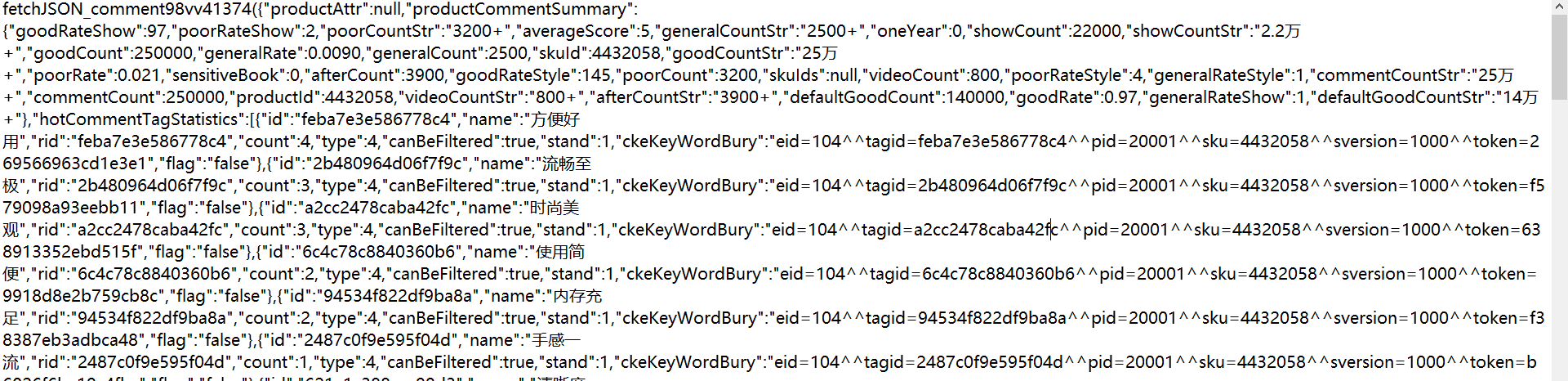
笔记本一台

1. **实验内容与步骤（过程及数据记录）**
2. **数据采集**

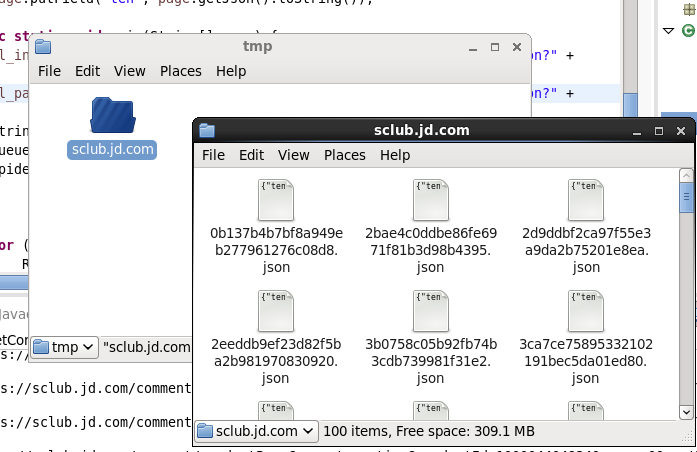
1.1 进入网址<https://item.jd.com/4432058.html>，查看浏览器请求的数据文件，在network->filter中输入pro选择productPageComments...文件，可以看到文件内容如下图所示：



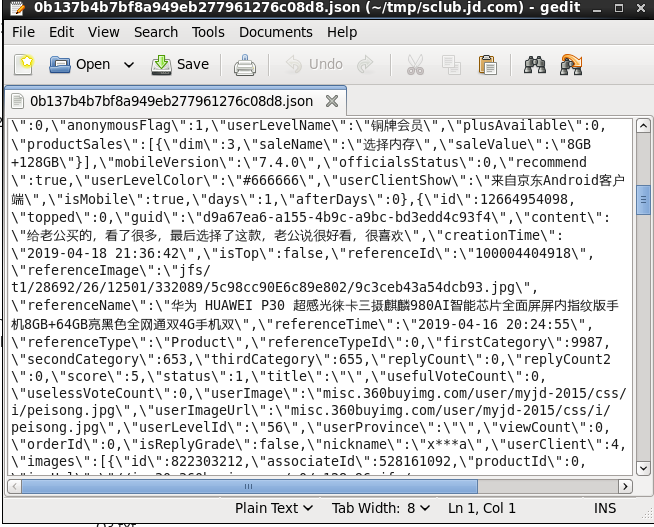
1.2 选中文件，右键复制call back在浏览器中打开，查看返回的Json形式的评论信息：



1.3 新建java工程，以java project的形式运行爬虫代码，查看使用java爬虫下载的数据是否成功、正确：



从上图可以看到确实是下载了100个文件，随即打开一个文件查看内容：

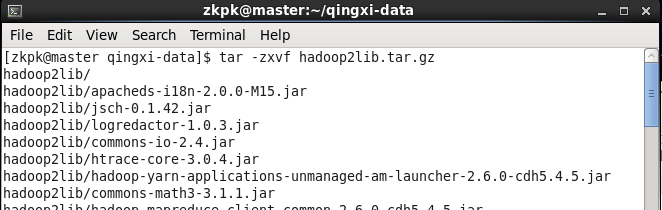


1.4 本次评论所使用的爬虫代码，更改了代理等设置：

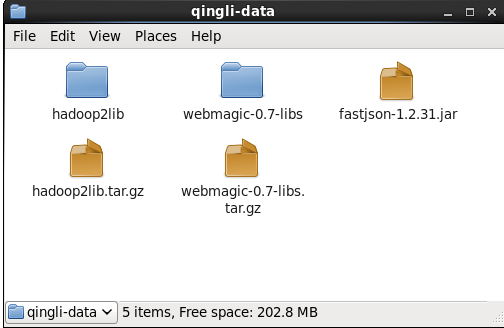
|  |
| --- |
| public class GetComments implements PageProcessor {  // 对爬取站点的一些属性进行设置，例如：设置域名，设置代理等；  private Site site = Site.*me*()  .setDomain("sclub.jd.com")  .setSleepTime(2000)  .addCookie("TrackID", "15yCkQ1-EBrUjdqR00DQ5QscyENaUom43JpYSvjMJpq6anA50TrPtspNLl6U8XaLsrVJcwNmEdABTf0HNRnM0w2bDvqpTE7RZbUMrf46d7Y0")  .addCookie("\_\_jda", "122270672.15289023667021425393993.1528902367.1528938475.1528969012.4")  .addCookie("\_\_jdb", "122270672.7.15289023667021425393993|4.1528969012")  .addCookie("\_\_jdc ", "122270672")  .addCookie("\_\_jdu", "15289023667021425393993")  .addCookie("\_\_jdv", "122270672|cps.youmai.com|t\_1000049399\_39857496|tuiguang|aed1c4304f694e599245a430daeddb2f|1528905344545")  .addHeader("User-Agent", "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.3")  .addHeader("Accept", "\*/\*")  .addHeader("Accept-Encoding", "gzip, deflate, br")  .addHeader("Accept-Language", "en-US,en;q=0.8")  .addHeader("Connection", "keep-alive")  .addHeader("Referer", "https://www.jd.com/")  .setUserAgent("Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.3");    public Site getSite() {  return site;  }  public void process(Page page) {  page.putField("ten", page.getJson().toString());  }  public static void main(String[] args) {  String url\_init = "https://sclub.jd.com/comment/productPageComments.action?" + "productId=100004404934&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold=1";  String url\_pattern = "https://sclub.jd.com/comment/productPageComments.action?" + "productId=100004404934&score=0&sortType=5&pageSize=10&isShadowSku=0&fold=1&page=";  String output = "/home/zkpk/tmp/";  QueueScheduler scheduler = new QueueScheduler();  Spider spider = Spider.*create*(new GetComments()).addUrl(url\_init)  .setScheduler(scheduler)  .addPipeline(new JsonFilePipeline(output))  .addPipeline(new ConsolePipeline());  for (int i = 1; i < 100; i++) {  Request request = new Request();  request.setUrl(url\_pattern + i);  scheduler.push(request, spider);  }  spider.thread(50).run();  }  } |

1. **数据清洗搭建解析框架**

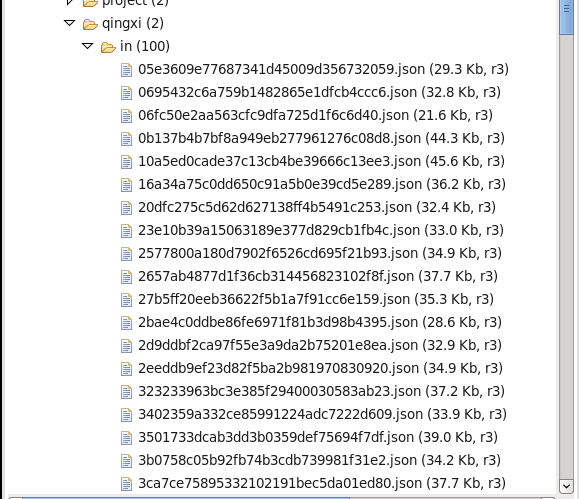
2.1 准备hadoop2lib.tar.gz和fastjson-1.2.31.jar软件包并解压到当前文件夹：



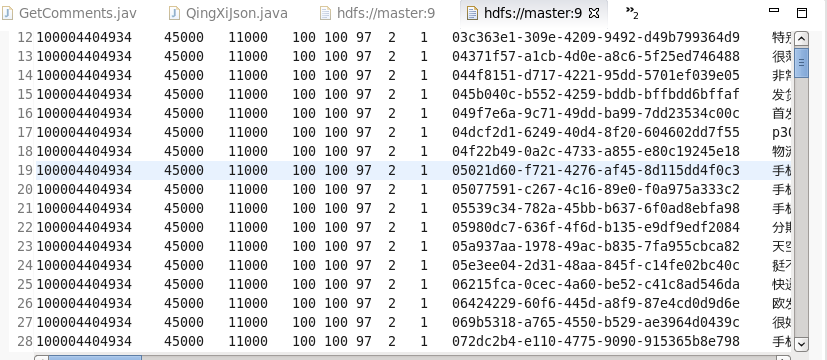
2.2 解压以及复制结果如下：



2.2 将所有评论上传到HDFS，并查看是否上传成功，此处为在eclipse中查看：



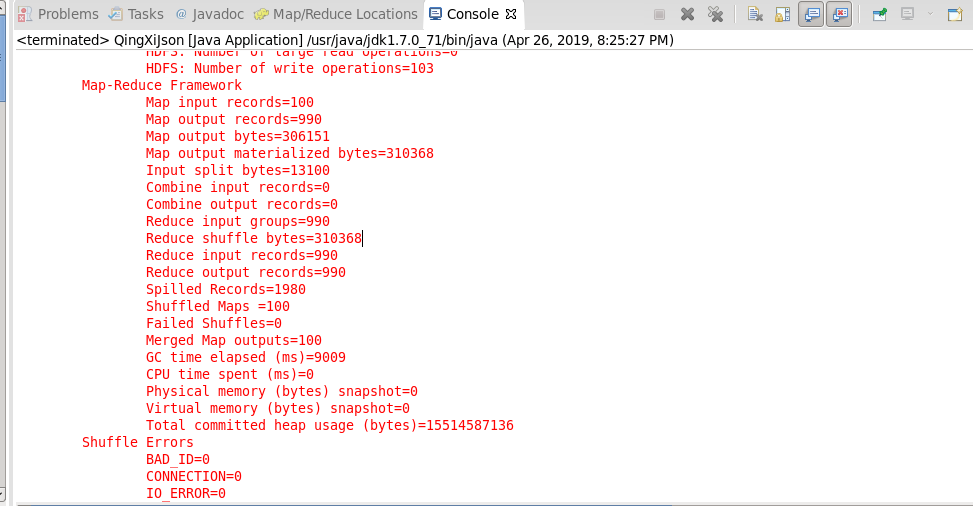
2.3 对HDFS中的评论进行清洗，将实用信息格式化并查看清洗结果：



2.4 将清洗结果从hdfs上下载到本地：



2.5 解决控制台不输出运行信息的问题，将~/hadoop-2.5.2/etc/hadoop/log4j.properties文件，拷贝到qingxi项目的src目录下。查看到输出结果，通过log日志排错，问题成功解决。

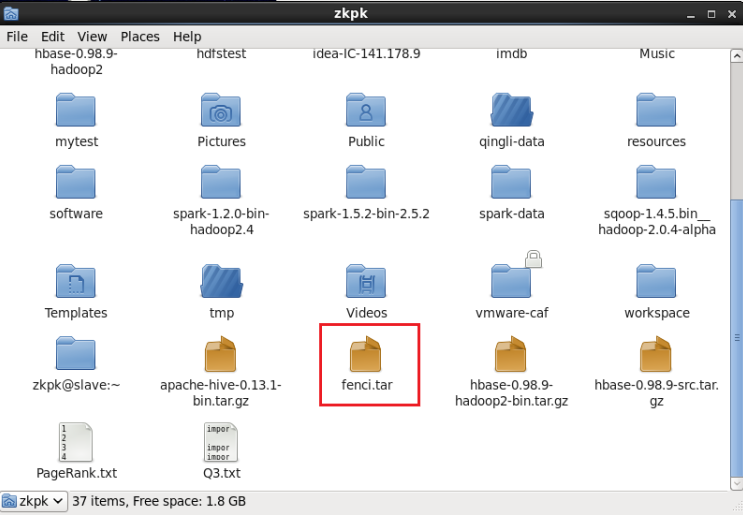


2.6 数据清洗代码

|  |
| --- |
| package my.mr;  import java.io.IOException;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.Mapper;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  import com.alibaba.fastjson.JSONArray;  import com.alibaba.fastjson.JSONObject;  public static class doMapper extends Mapper<Object, Text, Text, Text> {  @Override  protected void map(Object key, Text value, Context context)  throws IOException, InterruptedException {  String initJsonString = value.toString();  JSONObject initJson = JSONObject.parseObject(initJsonString );  if (!initJsonString.contains("productCommentSummary") && !initJsonString.contains("comments")) {  return;  }    JSONObject myjson = initJson.getJSONObject("ten");    JSONObject productCommentSummary = myjson.getJSONObject("productCommentSummary");  String productId = productCommentSummary.get("productId").toString();  String commentCount = productCommentSummary.get("commentCount").toString();  String goodCount = productCommentSummary.get("goodCount").toString();  String generalCount = productCommentSummary.get("generalCount").toString();  String poorCount = productCommentSummary.get("poorCount").toString();  String goodRateShow = productCommentSummary.get("goodRateShow").toString();  String generalRateShow = productCommentSummary.get("generalRateShow").toString();  String poorRateShow = productCommentSummary.get("poorRateShow").toString();  /\* comments 包括十条评论 \*/  JSONArray comments = myjson.getJSONArray("comments");  for (int i = 0; i < comments.size(); i++) {  JSONObject comment = comments.getJSONObject(i);  String guid = comment.getString("guid");  String content = comment.getString("content").replace('\n', ' ');  String creationTime = comment.getString("creationTime");  String score = comment.getString("score");  String nickname = comment.getString("nickname");  String userLevelName = comment.getString("userLevelName");  String userClientShow = comment.getString("userClientShow");  String isMobile = comment.getString("isMobile");  String days = comment.getString("days");  StringBuilder sb = new StringBuilder();    sb.append(productId); sb.append("\t");  sb.append(commentCount); sb.append("\t");  sb.append(goodCount); sb.append("\t");  sb.append(generalCount); sb.append("\t");  sb.append( poorCount ); sb.append("\t");  sb.append( goodRateShow ); sb.append("\t");  sb.append( generalRateShow ); sb.append("\t");  sb.append( poorRateShow ); sb.append("\t");  sb.append( guid ); sb.append("\t");  sb.append( content ); sb.append("\t");  sb.append( creationTime ); sb.append("\t");  sb.append( score ); sb.append("\t");  sb.append( nickname ); sb.append("\t");  sb.append( userLevelName ); sb.append("\t");  sb.append( userClientShow ); sb.append("\t");  sb.append( isMobile ); sb.append("\t");  sb.append( days );  String result = sb.toString();  context.write(new Text(result), new Text(""));  }  }  }  } |

1. **分词项目搭建**

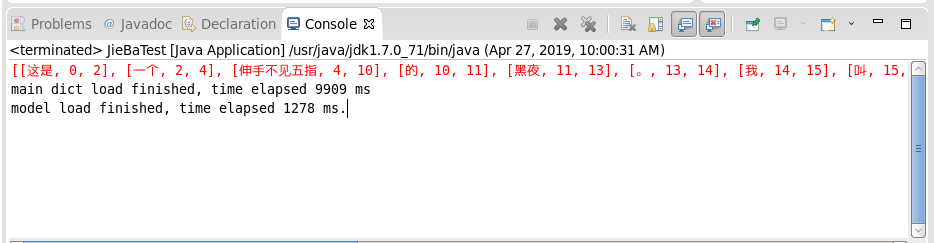
3.1 获取分词项目fenci.tar：



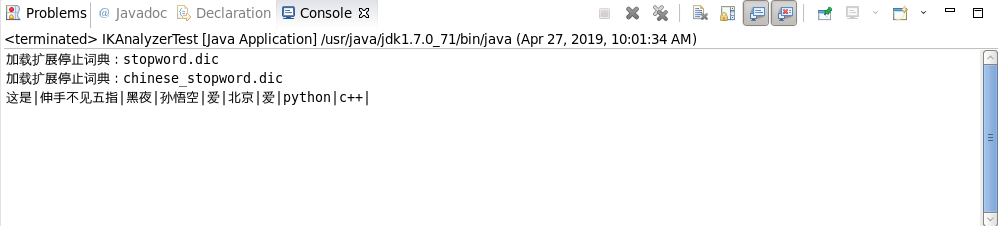
3.2 将fenci.tar导入到当前工程当中：



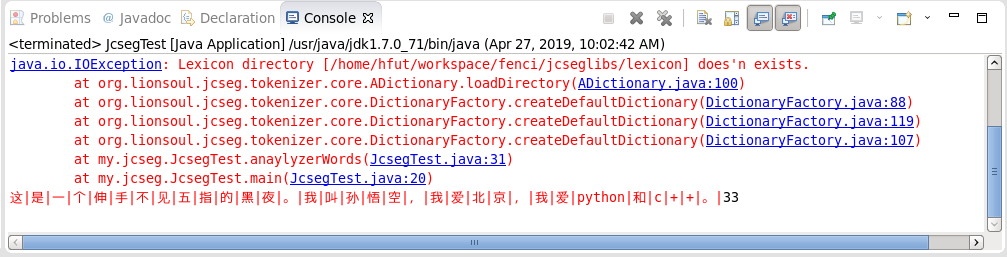
3.3 选择其中的jieBaTest.java文件并运行，查看结巴分词效果：



3.4 选择其中的IKAnalyzerTest.java文件并运行，查看IKAnalyzer分析效果：

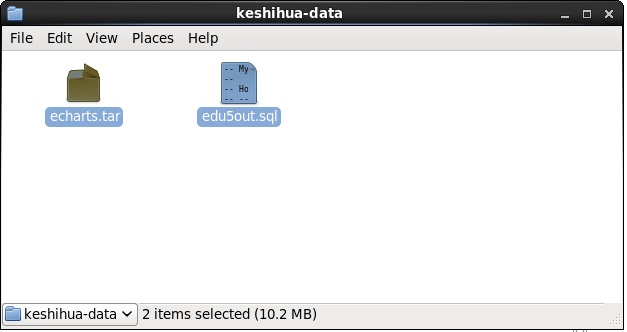


3.5 选择JcsegTest.java问价并运行，查看JcsegTest分词方法：



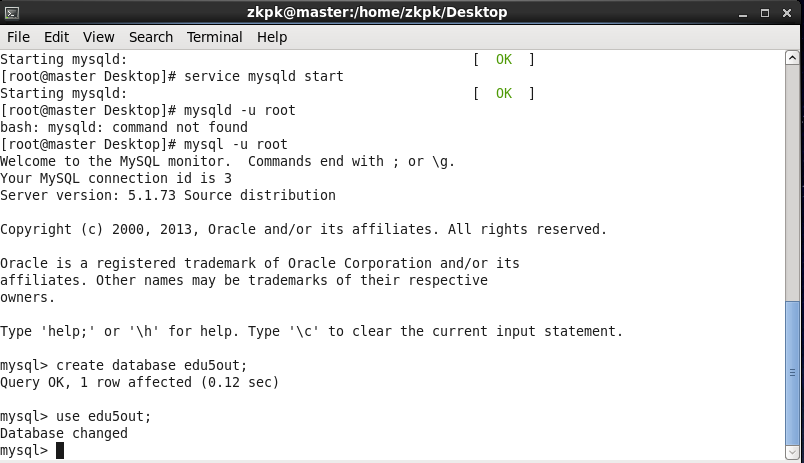
从以上结果中可以清楚的看到使用jieba和IKAnalyzer的分词效果要明显的优于第三个的分词效果，是否可以将人名、地名提取出来在一定程度上可以反应分词工具的好坏。

3.6 加载可视化文件，将echart.tar加载到eclipse中：



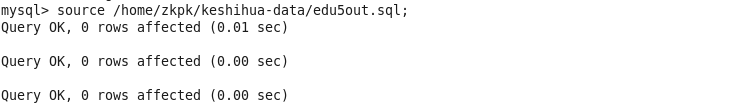
* 1. 使用service mysqld start命令打开mysql并创建新的数据库：

注意：此处使用的root登录。



1. **可视化**

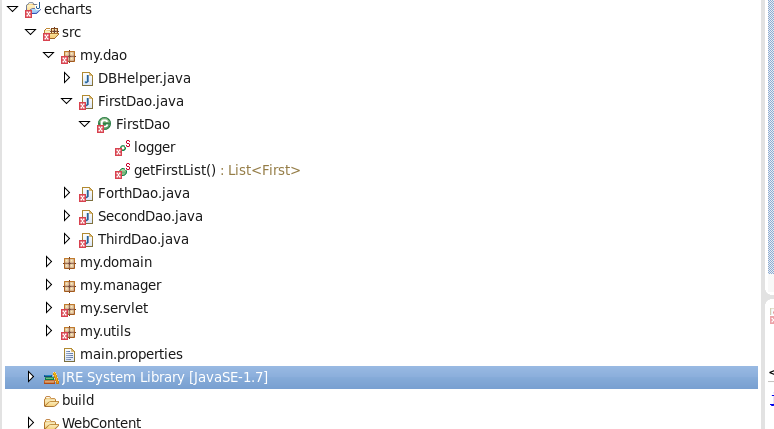
4.1 将/home/zkpk/keshihua-data目录下的edu5out.sql脚本导入到mysql的edu5out库中：



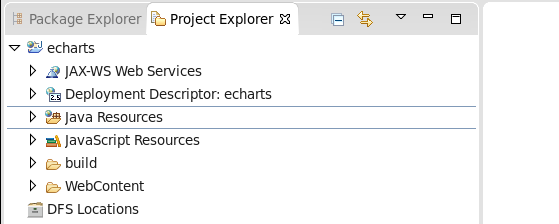
4.2 查看当前数据库（edu\_5out）中存在有哪些表（show tables）：



4.3 导入echart包后的项目情况：



4.4 修改jre system Library，重新配置之后的工程错误得到改正：



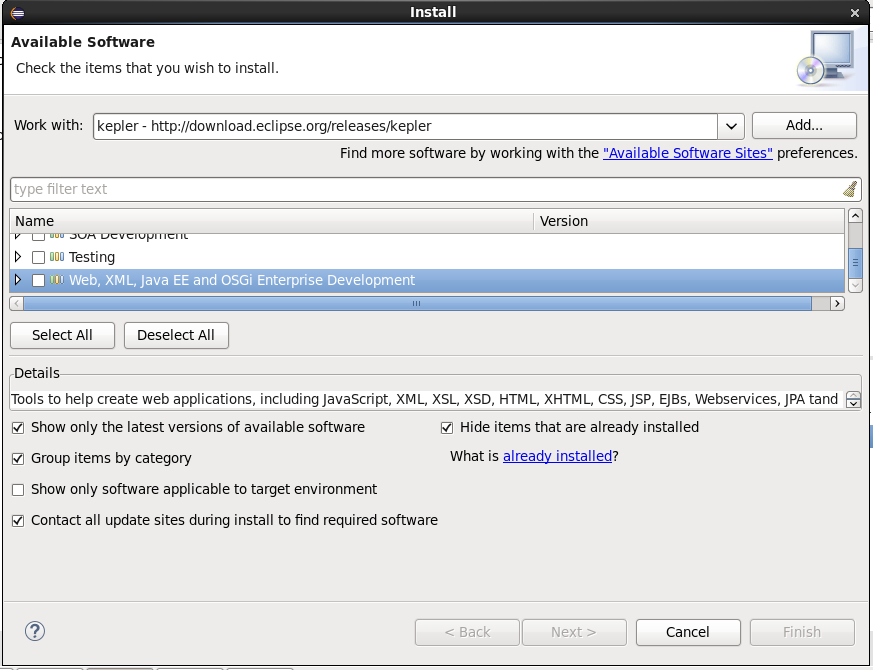
4.5 配置server（Tomcat）

4.5.1解决找不到server选项的问题：

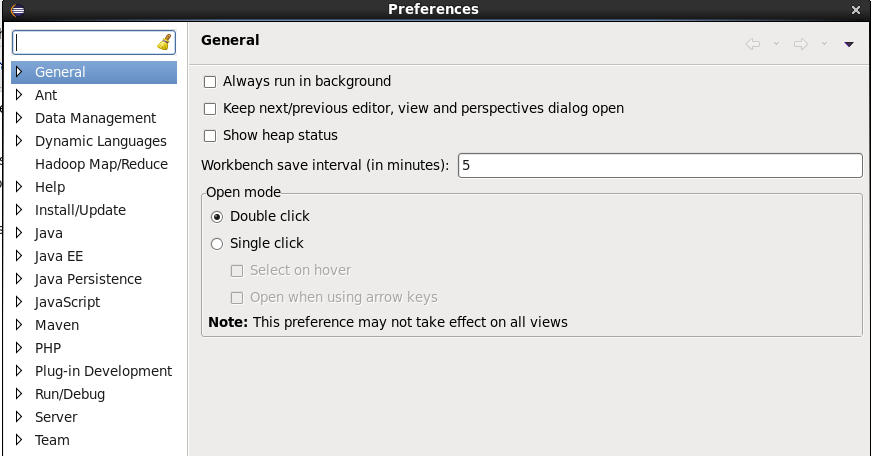
在work with中输入：

|  |
| --- |
| kepler - http://download.eclipse.org/releases/kepler |

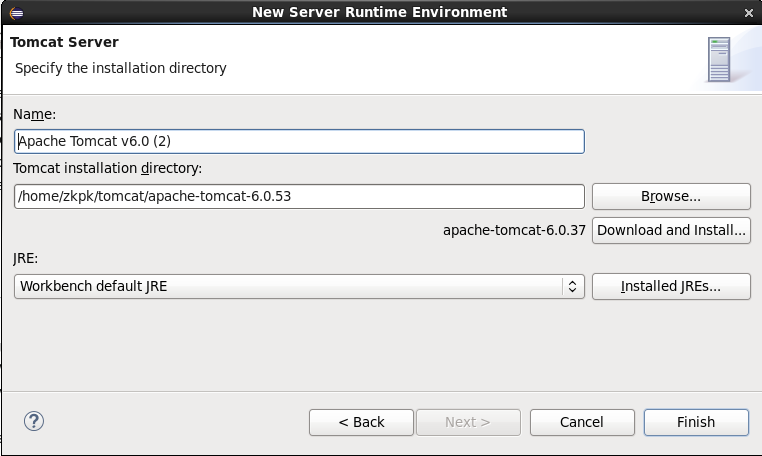
找到Web,XML, Java EE and OSGi Enterprise Development选项打钩，然后接下来所有选择next



4.5.2 在window->Preference中选择server：

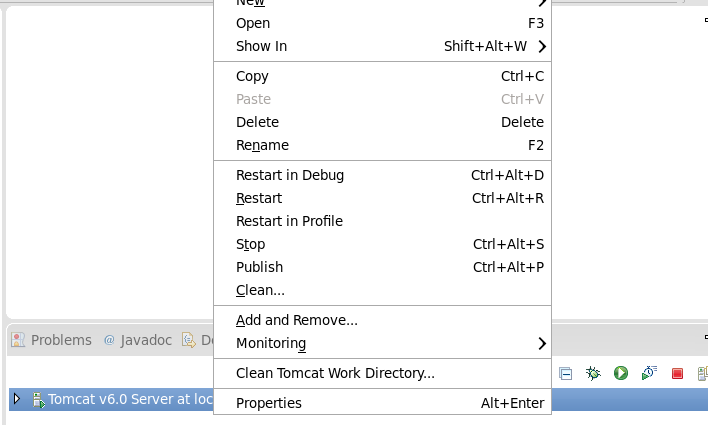


4.5.3 选择tomcat的安装路径：

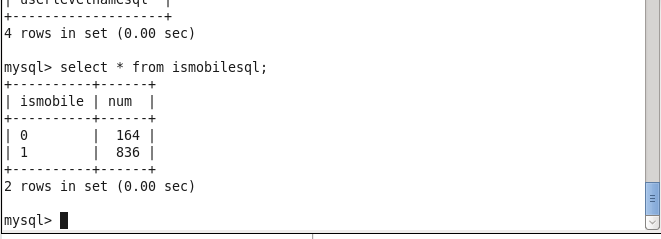


4.5.4 使server选项在下方任务栏显示，选中show view，在server中选中server选项，将其加入快捷方式：

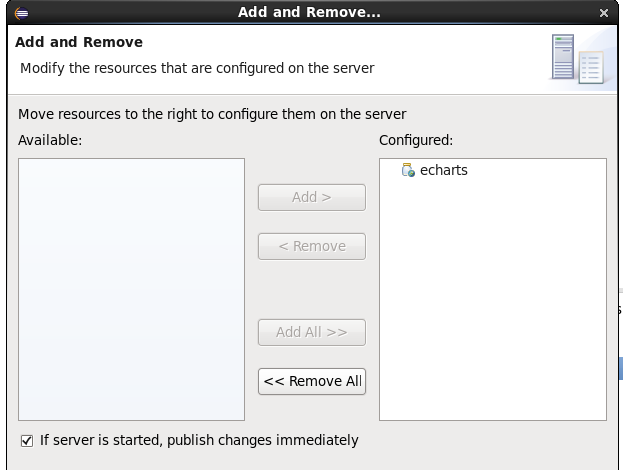
4.5.5 在下方任务栏中选中server，开启server（右键start）：



4.6 查看MySQL数据库中的ismobilesql表（select \* from ismobilesql;）其中1代表使用的是移动端，0代表使用的是PC端，可以看到两个数据分别为836和164：



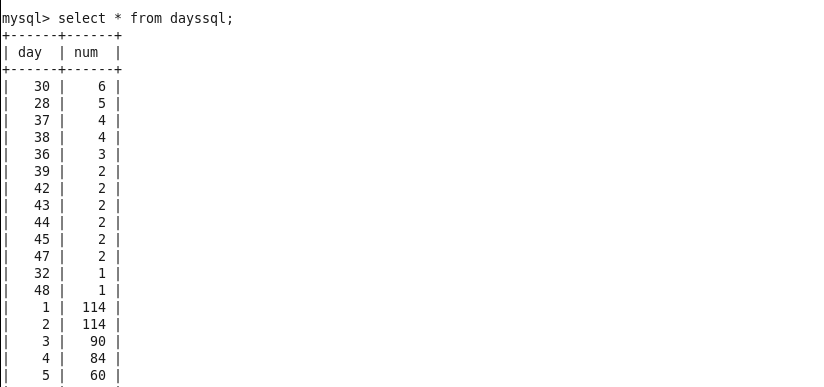
4.7 在下方的信息栏中选中server，右键选中Add and move将echart加入右方configured（使该项目在服务器中运行）：



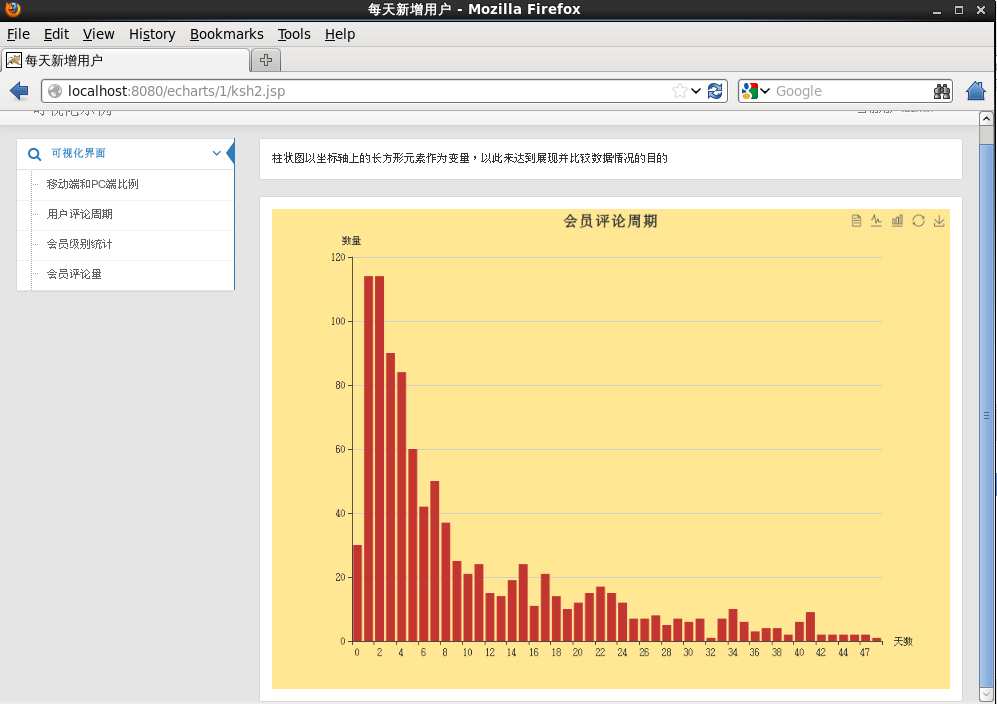
* 1. 在浏览器中输入：<http://localhost:8080/echarts/1/ksh1.jsp>查看效果，可以看到使用移动端和PC端的比例和在上方使用mysql查询到的数据是保持一致的，鼠标移动到饼图上面可以具体的看到每一部分的数量也是占比为836和164。

4.9 验证可视化比例是否正确，在mysql中输入下方语句，查看dayssql语句：

|  |  |
| --- | --- |
| select \* from dayssql; |  |

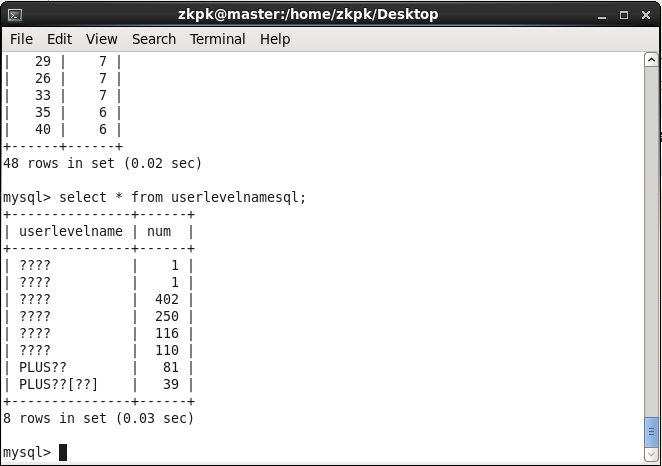


4.10 在浏览器中输入<http://localhost:8080/echarts/1/ksh2.jsp>查看每天的评论数量的直方图，显示效果如下图所示，可以看到和上面的使用mysql查出来的数据保持一致。



4.11 查看可视化结构是否正确，下图由于在创建数据库的时候编码错误因此出现乱码，在后面的过程中已经改正。

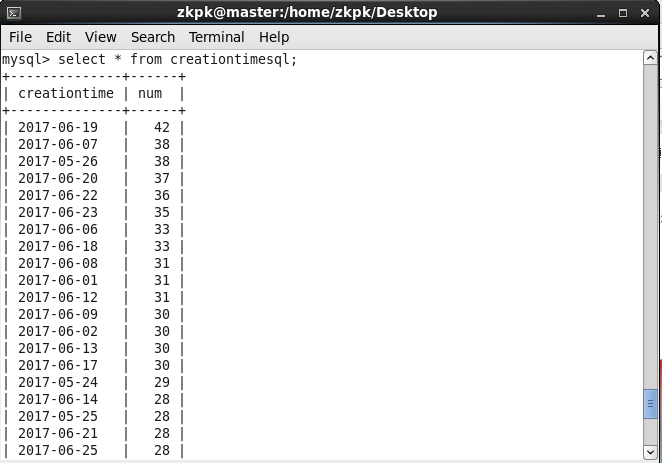
|  |
| --- |
| select \* from userlevelnamesql; |



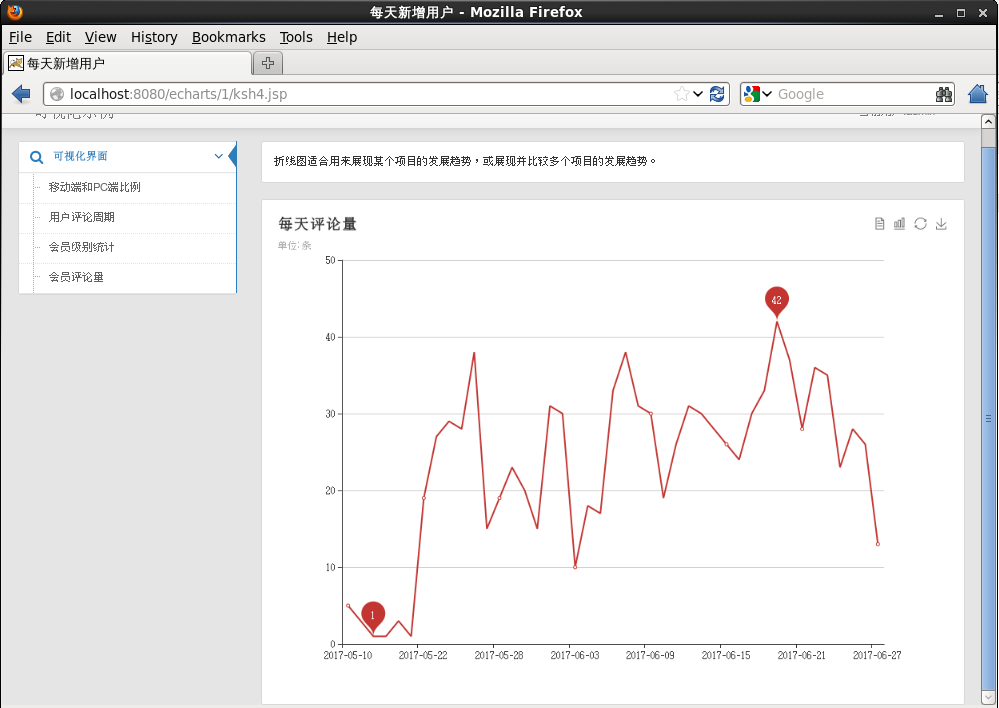
4.12 在浏览器中输入<http://localhost:8080/echarts/1/ksh3.jsp>查看各个等级会员所占的比例和上图一致。

4.13 使用mysql查看creationtime中的数据：

|  |
| --- |
| select \* from creationtimesql; |

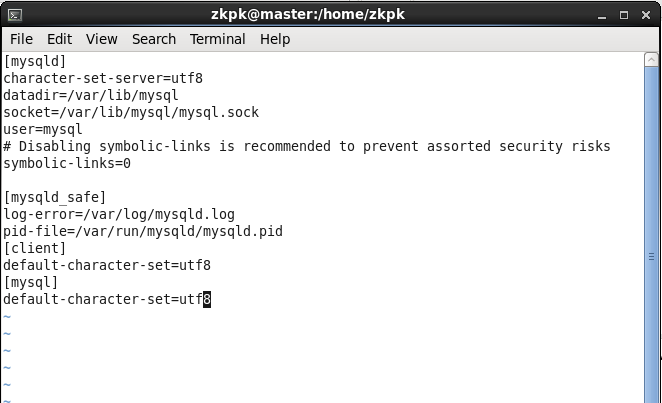


4.14 在浏览器中输入<http://localhost:8080/echarts/1/ksh4.jsp>查看每天的评论量。



1. **制作词云**

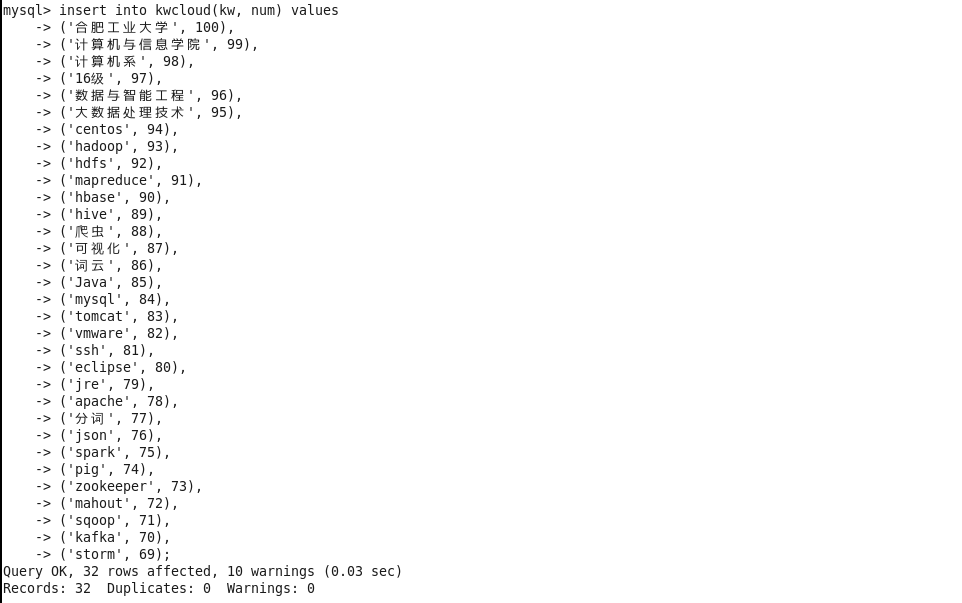
5.1配置字符编码，将数据库中的编码改为UFTF-8。



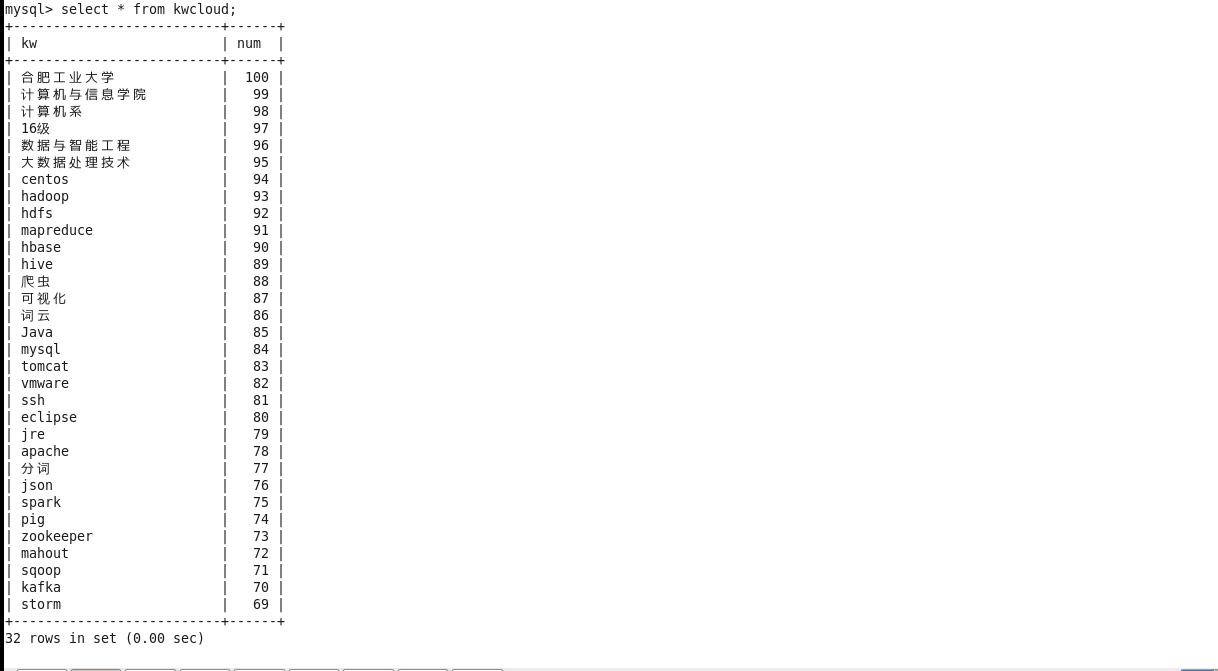
5.2重启mysql并建立数据库以及表：（注意在建立数据库的时候也要指定编码格式，不然在实验过程中仍然出错）。

|  |
| --- |
| create database ciyun default character set utf8 collate utf8\_general\_ci; |

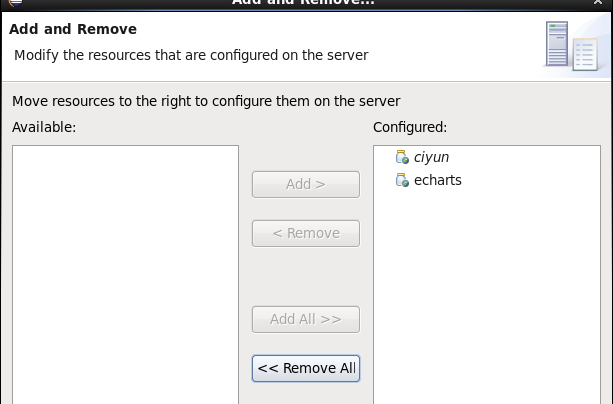
5.3 向kwcloud表中插入如下数据：



5.4 使用查询语句检查是否有乱码数据：



5.5 将词云项目加到configured中，使其在server中运行：



5.6 在浏览器中输入<http://localhost:8080/ciyun/1/ksh1.jsp>查看词云展示。

