基于大数据技术的招聘数据分析系统

目录

一、项目实现技术栈

本次项目使用了 hadoop 集群作为数据清洗的环境，其中 hadoop 集群包括 hadoop、zookeeper、hive、hbase 等软件组成。Tomcat 作为前端页面展示环境。数据清洗中，使用 java 编写的 MapReduce 进行特殊清洗，使用 hive 的 HQL 语句进行细化清洗，使用 java 编写的 UDF 进行特殊细化清洗吗，最后编写整合后的 shell 脚本来实现自动化清洗过程。前端页面展示中，使用 java 编写的 servlet 与 hbase 交互，获取数据并用 js 解析，把结果通过使用 echarts 表格、layui 等界面等框架配合优化展示效果，最终展示形式采用网页形式，展示网页中有一个交互小功能，该功能通过接收用户输入的数据进行权重匹配算法输出用户想要的内容。

二、项目实现思路

1.环境搭建

首先确定离线数据清洗环境，各个软件之间的兼容性，依赖包等等，下载对应文件后上传至 linux 中开始安装，具体步骤将会在环境搭建文档中展示。Hadoop 集群生态圈视图如图 2.1 所示。

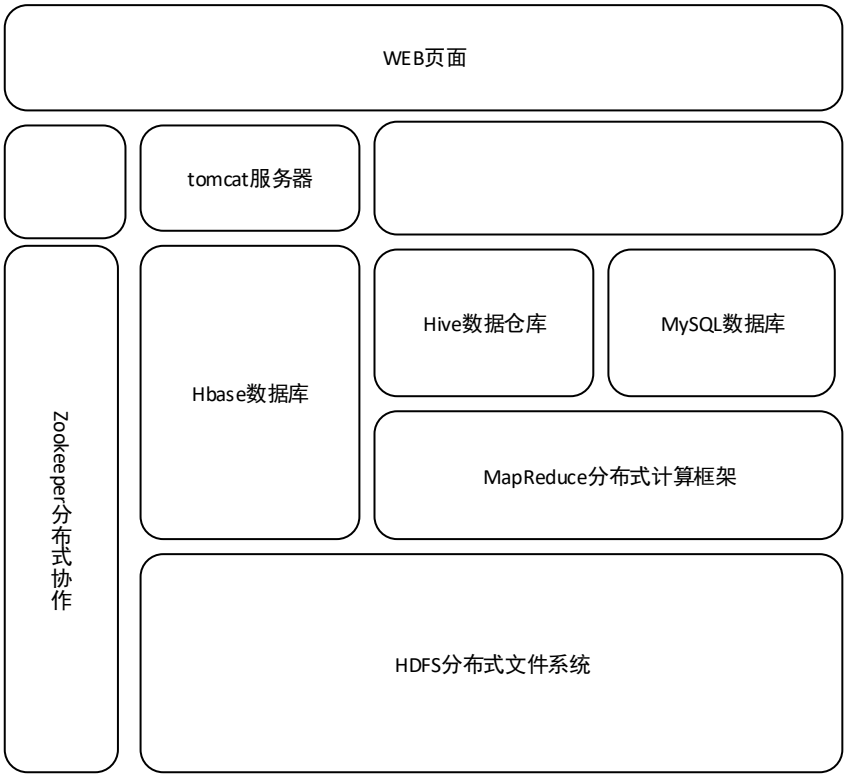

图 2.1Hadoop 集群生态圈

2.初步处理

在环境搭建完毕后开始数据处理阶段，先查询相关资料，如：csv 文件格式信息，hadoop 的使用说明，hive 清洗条件，hbase 与 hive 关联等等。根据

资料把数据全部装载到 hdfs 文件系统上并且载入到 hive 表格当中，在处理途中要确保数据正确没有打乱，并且过滤无用数据。

首先把 csv 清洗过滤一遍，这其中需要注意 csv 中数据的杂乱，需要应付各种格式情况，并且创建唯一标识符来当做 hvie 用来分隔列的标识符，这次采用键盘上无法输入的符号'\u0001'来当做分隔符，并且需要把这其中的无用数据过滤掉，这需要编写相应的 MapReduce 进行操作。当所有数据经过编写的 MapReduce 清洗过后，在 hive 中创建对应表格，并命名为 jobs 表。通过相应命令（load）把 hdfs 上被清洗过的文件加载到 hive 表格当中，并且途中反复效验数据是否合格。初步处理流程图如图 2.2 所示。

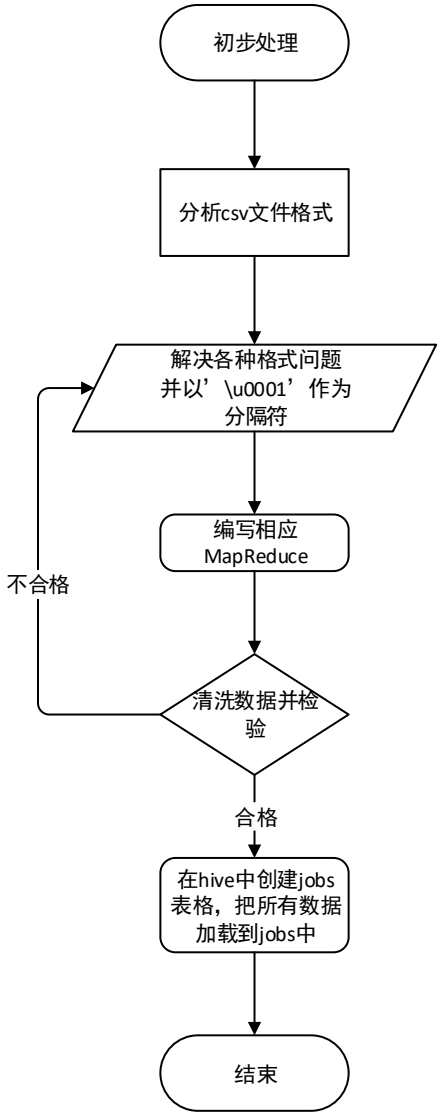

图 2.2 初步处理流程图

3.细化清洗

在所有数据载入到 hive 表格中后，对其进行更细化的、针对性的清洗，如：热门职业所需要掌握的技能，薪资排行前十的职业等等，根据所需要的内容，定制相应代码进行实现，并且在这过程中使用 HQL+UDF 的形式进行细化清洗，并把结果通过 hive 表与 hbase 的关联保存至 hbase 中。

首先确定需要达到的目标，并在 hive 中创建相应的表格，并且把它关联到 hbase 中，如 h_top_tag_gjc(热门职业所需要掌握的技能)、h_tag_salary_top（薪资排名前十的职业)、h_city_tag_salary（城市 IT 职业信息）等，要注意的是在创建表的时候要与 hbase 关联，这样可以省去 hive 迁移到 hbase 麻烦。确定目标后，编写相应的 HQL 语句，在这过程中，发现有些操作 HQL 无法实现，如对 job_info 中的技术词进行分词过滤，这时就需要编写相应的 UDF 来辅助 HQL 进行数据清洗。在清洗过程中，尽量不要把数据分开，数据尽量在同一张表内，并且有相应的逻辑关系，所以要合理利用相应的函数，如："split"、"explode"、"lateral view"等等。在清洗完毕后必须根据资料和常识等效验数据是否符合。细化清洗流程图如图 3.4 所示。

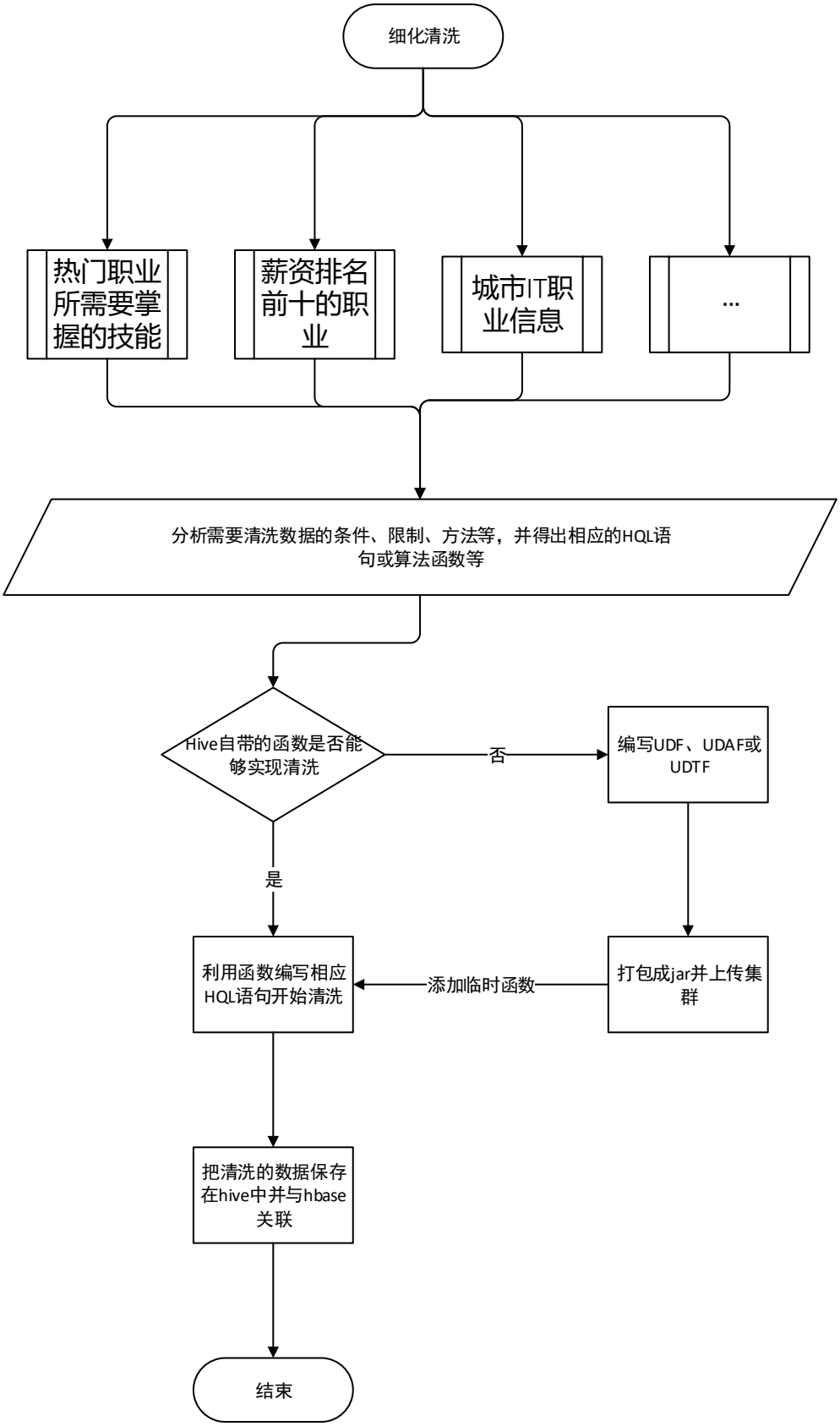
细化清洗
热门职业所需要掌握的技能
薪资排名前十的职业
城市IT职业信息
...
分析需要清洗数据的条件、限制、方法等，并得出相应的HQL语句或算法函数等
Hive自带的函数是否能够实现清洗
否
编写UDF、UDAF或UDTF
是
利用函数编写相应HQL语句开始清洗
添加临时函数
打包成jar并上传集群
把清洗的数据保存在hive中并与hbase关联
结束

图 3.4 细化清理流程图

4.与前端交互并展示

在一切数据处理好之后，创建 web 项目，因为访问 hbase 有延迟，不能实时与前端交互，本次采用服务端的定时任务，在 tomcat 启动时访问 hbase 获取所需的所有数据，并打包成 json 格式，以便前端能够实时访问数据。前端在获取到数据后，对其进行解析并把数据放入到相应的数据表格当中显示，在前端页面中，信息总量在页面左上角，还有八块信息显示区域，分别是：“职业关键词”、“招聘信息展示”、“招聘数据占比”、“地区薪资”、“IT 均薪 TOP10”、“热门职位 TOP5”和“城市招聘信息情况”，分布在页面各个区域，并无先后主次之分，这些显示区域大多使用 echarts 框架表格，以达到展示目的；一块功能区域“点击测试适合你的工作岗位”，该功能是一个简单的问卷调查，根据提交信息推送工作岗位消息。

完成以上功能首先解决前后端数据传输问题，创建定时任务，在 tomcat 启动时便开始执行访问 hbase 的代码，并把访问结果分类打包成 json 保存在内存中（静态变量），方便前端实时调用。前端使用 ajax 请求到 json 后，分别解析，把所有数据放入到页面的各个相应的表中，确保数据准确、不乱码、数据完整等展示效果。前端交互展示流程图如图 2.5 所示。

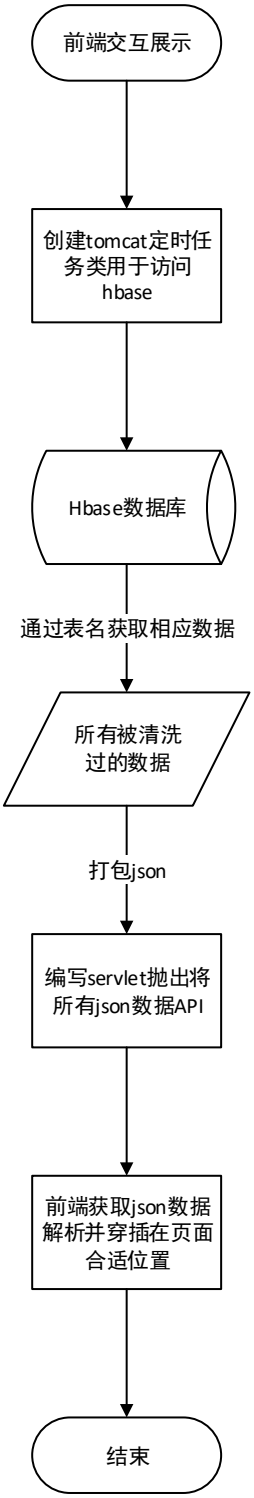

图 2.5 前端交互展示流程图

三、项目实现流程介绍

1.清洗数据

在集群环境搭建完毕之后（环境搭建流程参考环境搭建文档），开始第二阶段清洗数据。

1）上传数据

首先通过“Xftp”软件把六个文件（jobs1.csv~jobs6.csv）上传至 linux 的一个 data 文件内，进入到该文件夹，执行以下命令。

① hadoop fs -mkdir /data

② hadoop fs -put * /data

把数据上传至 hdfs 的一个 data 文件夹内。

2）过虑数据并添加分隔符

在观察这六个数据文档后，我们判断 hive 无法再数据不打乱的情况下正常加载数据文件，便使用 java 编写相应的 MapReduce 来辅助 hive 载入文件到数据库内。

在该 MapReduce 中，需要实现以下功能：

A． 要完美解析 csv 格式文件，通过 csv 格式文件的特点对他进行分列并

替换分隔符。

B. 注意文件格式紊乱,要考虑多种情况,比如某条数据列数不足标准(14列)、某条数据换行导致数据跨越维度等等。

C. 过滤掉无效数据,比如 job_name、job_info 为空或者为 null,job_name 中含有“实习”标签的数据。

相应的 MapReduce 代码和用法将在第五节项目代码介绍中详细展示。

在编写完毕后，打包成 jar 上传到 linux 中，在 linux 下执行以下代码把六个在 hdfs 中的 csv 文件全部转化成所需要的格式文件。

① hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs1.csv /out/t1

② hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs2.csv /out/t2

③ hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs3.csv /out/t3

④ hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs4.csv /out/t4

⑤ hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs5.csv /out/t5

⑥ hadoop jar xie-mr1.jar test.mapreduce.t2.Main /data/jobs6.csv /out/t6

3）把数据加载至 hive 表格中

在数据已经可以被 hive 表识别之后，在 hive 中创建招聘数据总表，并把 hdfs 中的被清洗过后的数据文件加载到该表格中。

① create table jobs_spys(company_financing_stage string,company_industry string,company_location string,company_name string,company_nature string,company_overview string,company_people string,job_edu_require string,job_exp_require string,job_info string,job_name string,job_salary string,job_tag string,job_welfare string) row format delimited fields terminated by '\u0001';

② load data inpath "/out/t1/part-r-00000" into table jobs_spys;

③ load data inpath "/out/t2/part-r-00000" into table jobs_spys;

④ load data inpath "/out/t3/part-r-00000" into table jobs_spys;

⑤ load data inpath "/out/t4/part-r-00000" into table jobs_spys;

⑥ load data inpath "/out/t5/part-r-00000" into table jobs_spys;

⑦ load data inpath "/out/t6/part-r-00000" into table jobs_spys;

2.针对性清洗

所有数据都已确保无误存储在 hive 的表格中，可以进入到 hive 中对其进行更细化的针对性清洗。

1）准备性工作

在针对性数据清洗前，需要对针对数据进行分析和查阅资料，编写出相应的 HQL 语句进行测试，在这过程中，我们发现某些作业中使用 hive 自带的函数无法实现某些清洗数据时的特殊需求，这时需要编写 UDF、UDTF 和 UDAF 来辅助完成作业。这些特殊作业中，“前十职业的技能关键词”需要对 job_info 进行分词并过滤掉非技术词；“均薪排行前十职业”需要对 job_salary 中的薪资描述合并成一个标准单位的数字（本次项目采用元/月单位）；各大“城市招聘信息”中需要对 company_location 进行分词归并成一种类型城市单位（本次项目采用除直辖市和自治区外全部以省为单位，如南昌市转化为江西省，上海市转化为上海市）。

确定以上需求后，编写特定的算法并组合成 UDF，最后打包成 jar 上传至 hadoop 集群中方便 hive 调用，其中 UDF 的关键代码和使用方法在第五节的项目代码介绍中详细展示。

进入到 hive，添加 jar 后添加 hive 的临时函数并命名，命令如下。

```
① add jar /usr/hadoop/jar/xie-udf2.jar;
② create temporary function fenci as
   "test.hive.udf.t1.HelloUDF";
③ create temporary function fenci_city as
   "test.hive.udf.t1.CityUDF";
④ create temporary function fenci_salary as
   "test.hive.udf.t1.SalaryUDF";
```

2）查询数据总量

查询过滤后的所有数据总量

```
① SELECT COUNT(job_name) FROM jobs_spys;
```

查询结果

192894

3）查询排行前十的职业

查询排行前十的职业信息

```
① select job_tag,count(1) as c from jobs_spys group by job_tag
   order by c desc limit 10;
```

查询结果

数据分析　　54525

数据挖掘　　29440

人工智能　　28370

机器学习　　23449

深度学习　　17871

算法　　15440

算法工程师　　7649

数据分析师　　6571

自然语言处理　　4904

机器视觉　　1276

4）清洗前十职业的技能关键词

针对实现思路中的“热门职业所需掌握的重要技能”，从总数据中清洗出相应数据，并把结果储存在 hbase 中，方便以后调用，该作业中涉及到分词操作，使用之前添加的临时函数 fenci 进行辅助清洗。以下是 HQL 语句。

① CREATE TABLE h_top_tag_gjc_spys
(job_tag string,gjc string,count string)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,info:job_info,info:count")

```
TBLPROPERTIES ("hbase.table.name" =
"h_top_tag_gjc_spys",
"hbase.mapred.output.outputtable" =
"h_top_tag_gjc_spys");
② insert into  h_top_tag_gjc_spys
③ select a.job_tag,concat_ws(',',collect_set(a.gjc)) as
gjc_z,concat_ws(',',collect_set(cast(a.count as
string))) as count_z from
(select gjc,job_tag,count(1) as count from (select
job_tag,gjc from  jobs_spys
lateral view explode(split(fenci(job_info),','))t1 as
gjc)b where b.job_tag='数据分析' or    job_tag='数据
挖掘' or    job_tag='人工智能' or    job_tag='机器学
习' or    job_tag='深度学习' or    job_tag='算法'
or    job_tag='算法工程师' or    job_tag='数据分析师'
or    job_tag='自然语言处理' or    job_tag='机器视觉'
group by gjc,job_tag)a
group by a.job_tag;
```

清洗出 10 条结果，具体内容将在页面中对应模块会详细展示。排行前十职业清洗结果图如图 3.1 所示。

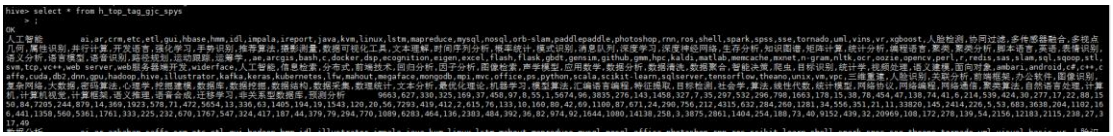

图 3.1 排行前十职业清洗结果图

5）清洗均薪排名前十职业

针对实现思路中的“均薪排名前十职业”，从总数据中清洗出相应数据，并把结果储存在 hbase 中，方便以后调用，该作业中涉及到薪水计算操作，使用之前添加的临时函数 fenci_salary 进行辅助清洗。以下是 HQL 语句。

① CREATE TABLE h_tag_salary_top_spys
(job_tag string,salary int)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,info:salary")
TBLPROPERTIES ("hbase.table.name" =
"h_tag_salary_top_spys",
"hbase.mapred.output.outputtable" =
"h_tag_salary_top_spys");

② insert into h_tag_salary_top_spys
select tag,cast(avg(salary) as int) as salary from
(select job_tag as tag,fenci_salary(job_salary) as
salary from jobs_spys)a

```
where salary!=0
group by tag
order by salary desc
limit 10;
```

清洗出 10 条结果，具体内容将在页面中美化后中详细展示。均薪排名前十职业清洗结果图如图 3.2 所示。

图 3.2 均薪排名前十职业清洗结果图

6）清洗城市招聘信息

针对实现思路中的“城市招聘信息”，从总数据中清洗出相应数据，并把结果储存在 hbase 中，方便以后调用，该作业中涉及到分析城市和分析工资操作，使用之前添加的临时函数 fenci_city 和 fenci_salary 进行辅助清洗。以下是 HQL 语句。

① CREATE TABLE h_city_tag_salary_spys

```
(city string,tag string,count string,salary int)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,info:tag,info:count,info:salary")
TBLPROPERTIES ("hbase.table.name" =
"h_city_tag_salary_spys",
"hbase.mapred.output.outputtable" =
"h_city_tag_salary_spys");
```

②
```
insert into h_city_tag_salary_spys
select
location,concat_ws(',',collect_set(job_tag)),concat_w
s(',',collect_set(cast(count as
string))),cast(avg(salary2) as int) from
(select fenci_city(company_location) as
location,job_tag,count(1) as count,avg(salary) as
salary2 from
(select
company_location,job_tag,fenci_salary(job_salary) as
salary from jobs_spys  )b
where salary!=0
group by company_location,job_tag)a
```

```
group by location;
```

清洗出 34 条结果，具体内容将在页面中美化后中详细展示。城市招聘信息清洗结果图如图 3.3 所示。

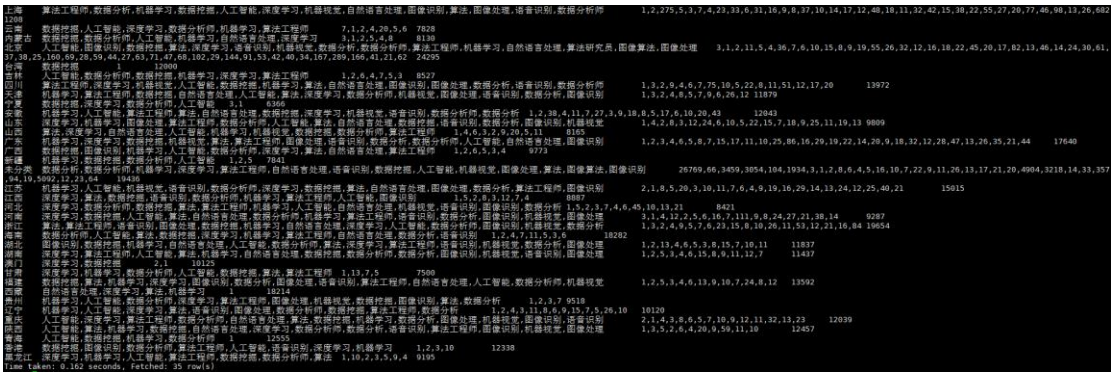

图 3.3 城市招聘信息清洗结果图

7）清洗招聘信息

针对实现思路中的“招聘信息”，从总数据中清洗出相应数据，并把结果储存在 hbase 中，方便以后调用，该作业中涉及到分析城市操作，使用之前添加的临时函数 fenci_city 进行辅助清洗，由于数据量太大，前端暂时只调用 2000 条数据。以下是 HQL 语句。

①
```
CREATE TABLE h_job_info_spys
(id int,city string,company_name string,job_name
string,salary string,job_tag string)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
```

```
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,info:city,info:company_name,info:
job_name,info:salary,info:job_tag")
TBLPROPERTIES ("hbase.table.name" =
"h_job_info_spys",
"hbase.mapred.output.outputtable" =
"h_job_info_spys");


②insert into h_job_info
select row_number() over (order by job_tag) as id,z.*
from
(
select distinct * from(
select
location,company_name,job_name,job_salary,job_tag
from
(select fenci_city(jobs.company_location) as
location,jobs.company_name,jobs.job_name,jobs.job_sal
ary,jobs.job_tag
from jobs,h_tag_top where
jobs.job_tag=h_tag_top.job_tag limit 700)a
```

```
union all
select
location,company_name,job_name,job_salary,job_tag
from
(select * from(select
fenci_city(jobs.company_location) as
location,jobs.company_name,jobs.job_name,jobs.job_sal
ary,jobs.job_tag
from jobs)bb,h_city_tag_salary where
bb.location=h_city_tag_salary.city limit 1300)b
)c
)z;
select
location,company_name,job_name,job_salary,job_tag
from
(select * from(select
fenci_city(jobs_spys.company_location) as location,
jobs_spys.company_name, jobs_spys.job_name,
jobs_spys.job_salary, jobs_spys.job_tag
from jobs_spys)bb, jobs_spys where bb.location=
jobs_spys.city limit 1300)b
)c
```

```
)z;
```

清洗出多条数据，具体内容在前端页面分析后展示。

8）清洗热门职业

针对实现思路中的“热门”，从总数据中清洗出相应数据，并把结果储存在 hbase 中，方便以后调用，以下是 HQL 语句。

①
```
CREATE TABLE h_tag_top_spys
(job_tag string,count int)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,info:count")
TBLPROPERTIES ("hbase.table.name" = "h_tag_top_spys",
"hbase.mapred.output.outputtable" =
"h_tag_top_spys");
```

②
```
insert into  h_tag_top_spys
select job_tag,count(1) as count from jobs_spys group
by job_tag order by count desc limit 5;
```

清洗出 5 条数据，具体内容将会被前端美化后展示。热门职业清洗结果图如

图 3.4 所示。

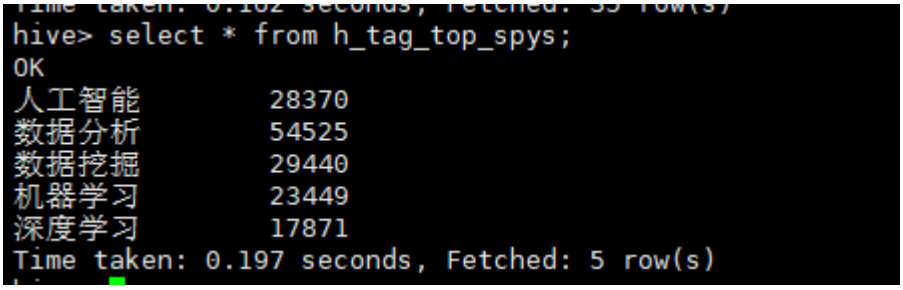

图 3.4 城市招聘信息清洗结果图

3.数据可视化处理展示

在所有需要数据离线清洗后，需要编写相应的服务端程序访问 hbase 后把数据传输给前端进行分析处理以及展示。服务端程序采用 java 编写的 web 应用，数据形式由服务端打包成 json 进行传输。前端将采用网页形式展示。

1）访问 hbase 并且打包成 json

在 web 应用引入 hbase 所需的依赖包后，创建一个类用于访问 hadoop 集群上的 hbase 数据库并包装成 json 格式。

编写 HbaseDemo 类来获取 hbase 数据库并且把所有数据打包成 json，由于 java 代码访问 hbase 数据时有延迟，不适合实时访问，所以采用 web 应用的定时任务技术，在 tomcat 启动时访问 hbase 中的数据，并把它打包成 json 储存在内存中（静态变量），最后创建一个 servlet 抛出 API 给前端提供数据，核心代码将会在第五节项目代码介绍中展示。

2）前端处理并展示

前端首先使用 ajax 请求到 json，并把数据解析分别放入到各个页面的各个区域中。数据解析后分别被八块区域利用，分别是“职业关键词”，“招聘数据占比”，“招聘信息展示”，“IT 行业均薪 TOP10”，“热门职业 TOP5”，“地区薪资”，“大数据招聘调查问卷”（隐藏），和中间的中国地图。在浏览器上访问“http://master2:8080/BigDataTest/index.html”，其中 master2 可以改为安装 tomcat 服务器的 ip 地址，最终展示效果图如图 3.5 所示。

图 3.5 最终展示效果图

点击右上角区域的“点击测试你合适的工作岗位”按钮可以打开大数据招聘调查问卷界面。大数据问卷调查界面图如图 3.6 所示。

图 3.6 大数据问卷调查界面图

四、项目分析结果说明

在上面的小节中，已经对项目数据分析的过程做了介绍，本小节将只介绍项目数据分析和结论。

1.数据总量分析

在对过滤后的数据进行总量查询后，得出职位名称和职位信息不为空同时不包含实习的职位名称的招聘数据是 192894 条。

2.招聘热度排行前十职业分析

对总数据根据 job_tag 进行分类并计数，得出热度排行前十的职业信息。前十职业信息数据如图 4.1 所示。

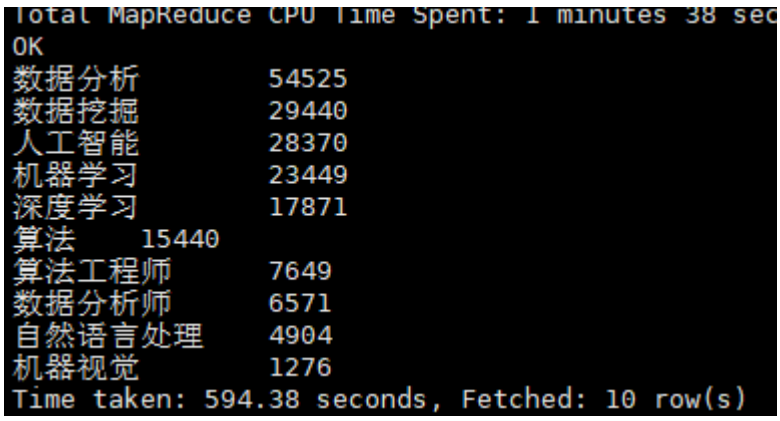

图 4.1 前十职业信息数据

3.前十职业技能关键词分析

①首先把 job_info 分词过滤并打散成一行一个关键词的状态，分词采用自己编写的 UDF 进行作业。

②通过 count 函数把所有关键词进行计数。

③通过 job_tag 把关键词和关键词数量进行整合，关键词列与关键词数量列中的数据是用逗号隔开并且是一一对应的关系。

④最后把热门职业前十的 job_tag 的数据保存在表中。

得出以下部分结果。前十职业关键词部分数据如图 4.2 所示。

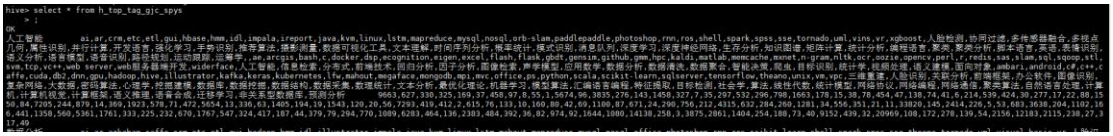

图 4.2 前十职业关键词部分数据

4.均薪排名前十职业分析

①首先对 job_salary 进行分词并求平均值，如 5k-8k 结果是 6500，分词并求平均值采用自己编写的 UDF 进行作业。

②过滤掉分词后薪资等于 0 的数据，并把剩余数据根据 job_tag 进行聚合求平均值，最终结果是以平均薪资的降序排序的前十条数据。

得出以下结果。均薪前十职业数据如图 4.3 所示。

```
hive> select * from h_tag_salary_top_spys;
OK
图像处理        23821
图像算法        29902
图像识别        27228
机器学习        27465
深度学习        26840
算法    36057
算法工程师      21445
算法研究员      31944
自然语言处理    41083
语音识别        25087
Time taken: 0.239 seconds, Fetched: 10 row(s)
```

图 4.3 均薪前十职业数据

5.城市招聘信息分析

①首先对首先对 job_salary 进行分词并求平均值。

②再对 company_location 进行城市分词，城市分词采用自己编写的 UDF 进行作业。并对 salary 根据城市求平均值并且对 job_tag 进行计数。

③最后根据城市把薪资数据和 job_tag 数据和 job_tag 数量进行整合，各个职业和职业数量之间使用逗号隔开。最后保存在表中。

得到以下结果。城市招聘信息数据如图 4.4 所示。

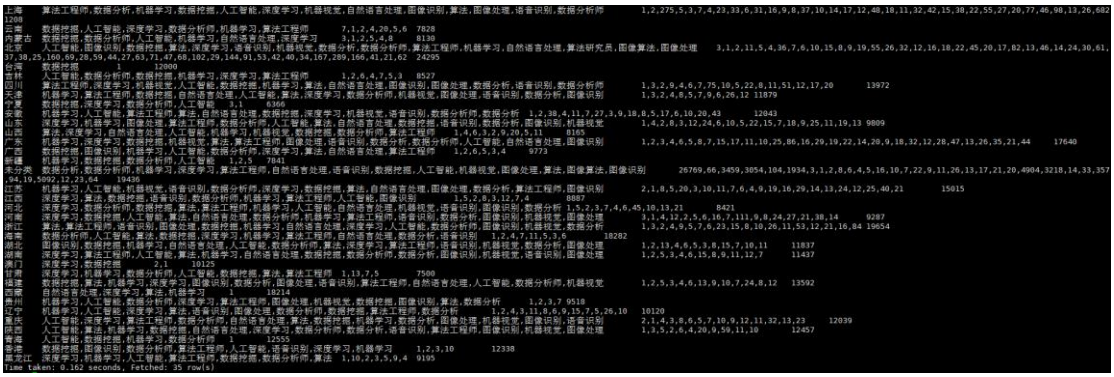

图 4.4 城市招聘信息数据

6.招聘信息分析

①首先对 company_location 进行城市分词，job_salary 进行薪资分词，再把相应的 company_name，job_name 数据进行整合，在这其中的两千条进行保存。

对其部分数据如下所示。招聘信息部分数据如图 4.5 所示。

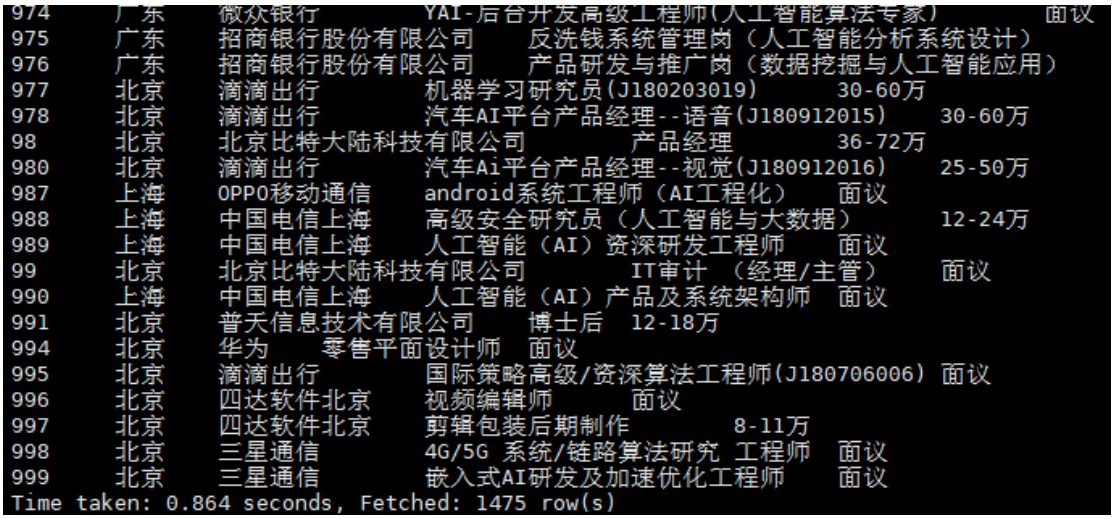

图 4.5 招聘信息部分数据

7.热门职业分析

①使用 count 函数通过 job_tag 标签对数据进行计数，并且把降序排序的前条保存在表格中。

结果如下。热门职业数据如图 4.6 所示。

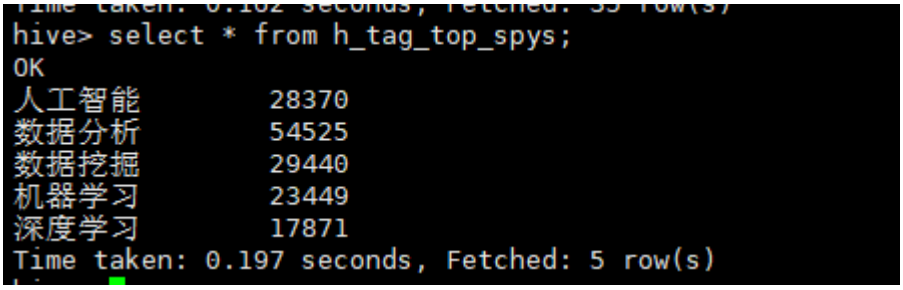
hive> select * from h_tag_top_spys;
OK
人工智能 28370
数据分析 54525
数据挖掘 29440
机器学习 23449
深度学习 17871
Time taken: 0.197 seconds, Fetched: 5 row(s)

图 4.6 热门职业数据

五、项目代码介绍

本小节主要对上文中提到过的代码进行展示和介绍。

1. 过虑数据并添加分隔符

1）csv 格式转换

以下是核心代码

①JobRinse 类：csv 格式分析算法

```
public class JobRinse {
   private Text T=new Text();
   private int Pdu(String str,int d) {
      int i=0,sum=0;
      if(!B) d++;
      while(true) {
         i=str.indexOf(",",d);
         if(i==-1) return -1;
         for(int k=i-1;k>=0;k--) {
            if(str.charAt(k)=='\"') sum++;
            else break;
         }
```

```
        if(sum%2==0) sum=0;
        else {
           B=false;
           return i;
        }
        d=i+1;
     }
  }
  private static int COUNT=0;
  private int COUNT2=0;
  private String[] STR=new String[14];
  private boolean B=false;
  public Text rinse1(Text value) {
      String str=value.toString();
      int count=1;
        for(int i=0;i<str.length();i++) {
           if(str.charAt(i)==',') count++;
        }
       String[] string=new String[count];
       COUNT2=0;
       int begin=0,end=0,k=0;
      while(k<count&&str.length()>0) {
```

```
            if(str.charAt(begin)    ==    '\"'    ||    B)
end=Pdu(str,end+2);
            else end = str.indexOf("," , begin );
            if(end < 0) end = str.length();
            string[k++] = str.substring(begin, end);
            begin=end+1;
            COUNT2++;
            if(end >= str.length() - 1) break;
         }
         boolean b=true;
         if(COUNT!=0) b=false;
         for(int i=0;i<COUNT2;i++) {
            if(!b) STR[COUNT-1]+=string[i];
            else {
               try {
               STR[COUNT]=string[i];
               }catch(Exception e) {
                  System.out.println("报错---------------
---------------------"+COUNT);
                  break;
               }
               COUNT++;
```

```
            }
            if(!b) b=true;
        }
        if(COUNT<13) {
            B=true;
            return null;
        }
        else {
        B=false;

if(STR[9]==null||STR[10]==null||"".equals(STR[10])||S
TR[10].contains("实习")){
            COUNT=0;
            return null;
        }
        StringBuffer sb=new StringBuffer();
        for(String s:STR) {
            sb.append(s);
            sb.append('\u0001');
        }
        COUNT=0;
        T.set(sb.toString());
```

```
        return T;
        }
       }
     }
```

②Mapeer1 类：mapeer 环节

```
public class Mapper1 extends Mapper<LongWritable,
Text, Text, NullWritable>{
      Text out;
      JobRinse jj=new JobRinse();
   protected void map(LongWritable key, Text
value,Mapper<LongWritable, Text, Text,
NullWritable>.Context context)throws
java.io.IOException, InterruptedException {
      out=jj.rinse1(value);
      if(out==null) return;
      context.write(out, NullWritable.get());
   }
    }
```

③Main 类：job 启动类

```
public class Main {
```

```
   public static void main(String[] args) throws
IOException, ClassNotFoundException,
InterruptedException {
      Configuration conf=new Configuration();
      Job job=Job.getInstance(conf);
      job.setJarByClass(Mapper1.class);
      job.setMapperClass(Mapper1.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(NullWritable.class);
      FileInputFormat.setInputPaths(job, new
Path(args[0]));
      FileOutputFormat.setOutputPath(job,new
Path(args[1]));
       try {
              job.waitForCompletion(true);
          } catch (ClassNotFoundException |
InterruptedException e) {
              e.printStackTrace();
          }
   }
    }
```

2）打包使用

把所有代码打包成 jar，并命名为 xie-mr1.jar，上传至 hadoop 集群通过包名.类名的方式调用并启动 job。

2.准备性工作

1）HelloUDF 类

针对 job_info 的分词 UDF，临时函数名称：fenci。

```
public class HelloUDF extends UDF {
   SliptW3 sliptw3=new SliptW3();
   StringBuffer sb=new StringBuffer();
   public Text evaluate(Text str) throws IOException {
      Text result=new Text();
      if(str==null) {
         result.set("");
         return result;
      }
      ArrayList<String> ci =
            sliptw3.sliptword(str.toString(),new
ArrayList<String>());
      for(int i=0;i<ci.size();i++) {
```

```
            sb.append(ci.get(i));
            if(i!=ci.size()-1) sb.append(",");
        }
        result.set(sb.toString());
        sb.setLength(0);
        return result;
    }
     }
```

2）CityUDF 类

针对 company_locationde 分词的 UDF，临时函数名称：fenci_city。

```
public class CityUDF extends UDF{
    SliptW3 sliptw3=new SliptW3();
    Text t=new Text();
    public Text evaluate(Text value) throws IOException
{
        Text result=new Text();
        if(value==null) {
            result.set("未分类");
            return result;
        }
```

```
        String str=sliptw3.Mcity(value.toString());
        if("".equals(str)) str="未分类";
        t.set(str);
        return t;
    }
     }
```

3）SalaryUDF 类

针对 job_salary 分词的 UDF，临时函数名称：fenci_salary。

```
public class SalaryUDF extends UDF{
    salary s=new salary();
    Text t=new Text();
    public Text evaluate(Text value) throws IOException
{
        Text result=new Text();
        if(value==null) {
            result.set("0");
            return result;
        }
        String str=s.count(value.toString())+"";
        t.set(str);
```

```
        return t;
    }
     }
```

4）SliptW3 类与 salary 类

这两个类是辅助前三个 UDF 的算法类，比如 HelloUDF 中使用的 sliptword 方法，CityUDF 中使用的 Mcity 方法，SalaryUDF 中使用的 count 方法。功能分别是职业计数词分词、城市分词以及计算均薪。

```
public class SliptW3 {
   public ArrayList<String> readtxt(ArrayList<String>
cibiao) throws IOException {
      InputStream
is=this.getClass().getResourceAsStream("cibiao.txt");
      InputStreamReader read1 = new
InputStreamReader(is, "utf-8");
      BufferedReader br = new BufferedReader(read1);
        String line="";
        line = br.readLine();
          while (line != null) {
              line = br.readLine();
              cibiao.add(line);
```

```
            }
            return cibiao;
    }
    public boolean Judge(String
string,ArrayList<String> cibiao) {
        for(int i=0;i<cibiao.size()-1;i++) {
            if(cibiao.get(i).length()<string.length()) {
                continue;
            }
            else {
                if(cibiao.get(i).substring(0,
string.length()).equals(string)) {
                    return true;
                }
            }
        }
        return false;
    }
    public ArrayList<String> delete(ArrayList<String>
cibiao,ArrayList<String> cibiao1){

        for(int i=0;i<cibiao1.size();i++) {
```

```
            if(!cibiao.contains(cibiao1.get(i))) {
               cibiao1.remove(i);i--;
            }
         }
         return cibiao1;

      }
      public ArrayList<String> sliptword(String
string,ArrayList<String> cibiao) throws IOException {
         cibiao=readtxt(cibiao);
         ArrayList<String> cibiao2=new
ArrayList<String>();
         int begin=0;String str="";int w=0;
         for(int i=1;i<=string.length();i++) {

      if(string.length()==i)str=string.substring(begin).t
oLowerCase();
            else str=string.substring(begin,
i).toLowerCase();

      if(str.charAt(0)>=65&&str.charAt(0)<=90||str.charAt
(0)>=97&&str.charAt(0)<=122) w=1;
```

```
        if(!Judge(str,cibiao)) {

            if(str.length()>1) {

                if(w==1&&str.charAt(str.length()-
1)>=65&&str.charAt(str.length()-
1)<=90||str.charAt(str.length()-
1)>=97&&str.charAt(str.length()-1)<=122) {
                    cibiao2.add(str);w=0;
                }
                else {
                cibiao2.add(str.substring(0,
str.length()-1));i--;
                }
            }
            else {
                cibiao2.add(str);
            }
            begin=i;
        }
        else if(string.length()==i) {
```

```
   if(str.length()==1&&w==1&&string.charAt(string.leng
th()-1)>=65&&string.charAt(string.length()-
1)<=90||string.charAt(string.length()-
1)>=97&&string.charAt(string.length()-1)<=122) {
                continue;
            }
            cibiao2.add(str);
         }
      }
      return delete(cibiao,cibiao2);
   }
   public ArrayList<String> readtxt(String str) throws
IOException {
      ArrayList<String> province=new
ArrayList<String>();
      InputStream
is=this.getClass().getResourceAsStream(str);
      InputStreamReader reader = new
InputStreamReader(is,"utf-8");
        BufferedReader br = new
BufferedReader(reader);
        String line="";
```

```
        line = br.readLine();
          while (line != null) {
              line = br.readLine();
              province.add(line);
          }
          return province;
    }
public String city(ArrayList<String> province,String
string) {

      for(int i=0;i<province.size()-1;i++) {
         String[] pro=province.get(i).split(",");
         for(int j=0;j<pro.length;j++) {
            if(string.contains(pro[j])) {
              return pro[0];
            }
         }
      }
      return "1";
    }
    public String Mcity(String str2) throws IOException
{
```

```
      ArrayList<String>
province=readtxt("province.txt");
      String str="1";
      str=city(province,str2);
            if(!str.equals("1"))
               return str;
            return "";
   }
    }

public class salary {
   public double Wrow(String string,int i) {
      string=string.substring(0, string.length());
      String[] str=string.split("-");
      double a=Double.valueOf(str[0]);
      double b=Double.valueOf(str[1]);
      switch(i) {
      case 1:return (a+b)/2*1000;
      case 2:return (a+b)/2;
      case 3:if(b<10) {return (a+b)/2*10000;}
             else {return (a+b)/24*10000;}
      case 5:return (a+b)/2*10000;
```

```
        case 6:return (a+b)/24*10000;
        case 7:return (a+b)/2*30;
        default :return (a+b)/2;
        }
    }
    public int Row(String string) {
        if(string.contains("K")) {
            string=string.replaceAll("K","");
            return (int)Wrow(string,1);
        }
        else if(string.contains("k")) {
            string=string.replaceAll("k","");
            return (int)Wrow(string,1);
        }
        else if(string.contains("元/月")) {
            string=string.replaceAll("元/月","");
            return (int)Wrow(string,2);
        }
          else if(string.contains("万")) {
            string=string.replaceAll("万","");
            return (int)Wrow(string,3);
        }
```

```
      else if(string.contains("千/月")) {
       string=string.replaceAll("千/月","");
       return (int)Wrow(string,1);
    }
      else if(string.contains("万/月")) {
       string=string.replaceAll("万/月","");
       return (int)Wrow(string,5);
      }
      else if(string.contains("万/年")) {
       string=string.replaceAll("万/年","");
       return (int)Wrow(string,6);
    }
      else if(string.contains("元/日")) {
       string=string.replaceAll("元/日","");
       return (int)Wrow(string,7);
      }
     else {
       return (int)Wrow(string,8);
      }
}
public int count(String string) throws IOException{
   try {
```

```
        return Row(string);
        }
        catch(Exception e) {
           return 0;
        }
    }
     }
```

5）打包使用

把所有代码打包成 jar 包，并命名为 xie-udf2.jar，上传至 hadoop 集群，在 hive 中使用 add jar 路径的命令进行添加 jar 包，使用 create temporary function 函数名 as “包名.类名”的命令添加临时函数。

3.数据可视化处理展示

1）HbaseDemo 类

HbaseDemo 类是获取 hadoop 集群上的数据库的数据并打包成 json 数据方便调用。

```
public class HbaseDemo {
   private boolean Judge(int s,int[] src,int k) {
      for(int i=0;i<k;i++) {
```

```
            if(s==src[i]) {
                return true;
            }
        }
        return false;
    }
    private int[] screen(int[] arr) {
    int[] scr=new int[10];int k=0;
    for(int i=0;i<scr.length;i++) {
        int max=0;int k1=0;
        for(int j=0;j<arr.length;j++) {
            if(!Judge(j,scr,k)) {
                if(arr[j]>max) {
                    max=arr[j];k1=j;
                }
            }
        }
        scr[i]=k1;k++;
    }
    return scr;
    }
     public static Configuration config;
```

```
        static {
            config = HBaseConfiguration.create();
            config.set("hbase.zookeeper.quorum",
"master2,s5,s6,s7");

config.set("hbase.zookeeper.property.clientPort",
"2181");
         }
     public String queryAllTable(String gjc,String
salary,String city,String tag_top,String
job_info_table)  {

            ResultScanner resultx=null;
            JSONObject jaResult=new JSONObject();
            try{
                System.out.println("---------------查询
gjc表数据 START-----------------");
                Connection  connection =
ConnectionFactory.createConnection(config);
                Table table =
connection.getTable(TableName.valueOf(gjc));
                resultx = table.getScanner(new Scan());
```

```
                JSONArray ja=new JSONArray();
                for (Result result : resultx) {
                    byte[] row = result.getRow();
                    System.out.println("row key is:" +
new String(row));
                    byte[] j =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("job_info"));
                    byte[] c =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("count"));
                    String[] job_info_tmp=new
String(j).split(",");
                    String[] sCount=new
String(c).split(",");
                    int[] count=new int[sCount.length];
                    String[] job_info=new
String[job_info_tmp.length];
                    for(int
i=0,jj=0;i<sCount.length;i++) {
                      if("".equals(job_info_tmp[i]))
continue;
```

```
                    job_info[jj]=job_info_tmp[i];

   count[jj]=Integer.parseInt(sCount[i]);
                    jj++;
                }
                JSONObject json=new JSONObject();
                json.put("name", new String(row));
                int[] xb=screen(count);
                for(int i=0;i<xb.length;i++) {
                  json.put(job_info[xb[i]],
count[xb[i]]);
                }
                ja.add(json);
                }
            jaResult.put("gjc", ja);
            System.out.println("---------------结束
gjc表数据 END-----------------");
            System.out.println("---------------查询
salary表数据 START-----------------");
            table =
connection.getTable(TableName.valueOf(salary));
            resultx = table.getScanner(new Scan());
```

```
                int[] s_salaryData=new int[10];
                String[] s_tagData=new String[10];
                int i=0;
                for (Result result : resultx) {
                    byte[] row = result.getRow();
                    System.out.println("row key is:" +
new String(row));
                    byte[] a =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("salary"));
                    s_salaryData[i]=Integer.parseInt(new
String(a));
                    s_tagData[i]=new String(row);
                    i++;
                    }
                int[] temp_xb=screen(s_salaryData);

                JSONObject json=new JSONObject();
                for(i=0;i<10;i++) {
                   json.put(s_tagData[temp_xb[i]],
s_salaryData[temp_xb[i]]);
                }
```

```
                jaResult.put("salary", json);
                System.out.println("---------------结束
salary表数据 END-----------------");
                System.out.println("---------------查询
city表数据 START-----------------");
                ja=new JSONArray();
                table =
connection.getTable(TableName.valueOf(city));
                resultx = table.getScanner(new Scan());
                for (Result result : resultx) {
                    byte[] row = result.getRow();
                    System.out.println("row key is:" +
new String(row));
                    byte[] a =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("tag"));
                    byte[] b =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("count"));
                    byte[] c =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("salary"));
```

```
                String[] tag_tmp=new
String(a).split(",");
                String[] sCount=new
String(b).split(",");
                int[] count=new int[sCount.length];
                String[] tag=new
String[tag_tmp.length];
                for(int
ii=0,jj=0;ii<sCount.length&&ii<tag_tmp.length;ii++) {
                  if("".equals(tag_tmp[ii]))
continue;
                  tag[jj]=tag_tmp[ii];

   count[jj]=Integer.parseInt(sCount[ii]);
                  jj++;
                }
                String s_tag="未分类";
                temp_xb=screen(count);
                for(i=0;i<temp_xb.length;i++) {
                  if(temp_xb[i]<tag_tmp.length) {
                     s_tag=tag_tmp[temp_xb[i]];
                     break;
```

```
                    }
                }
                String str=new String(row);
                if(!str.equals("未分类")) {
                JSONObject json2=new JSONObject();
                json2.put("name", new String(row));
                json2.put("variety",
tag_tmp.length);
                json2.put("main",s_tag);
                json2.put("salary",new String(c));
                ja.add(json2);
                }
                }
            jaResult.put("city", ja);
            System.out.println("---------------结束
city表数据 END-----------------");
            System.out.println("---------------查询
tag_top表数据 START-----------------");
            table =
connection.getTable(TableName.valueOf(tag_top));
            ja=new JSONArray();
            resultx = table.getScanner(new Scan());
```

```
                for (Result result : resultx) {
                    byte[] row = result.getRow();
                    System.out.println("row key is:" +
new String(row));
                    byte[] a =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("count"));
                    JSONObject json2=new JSONObject();
                    String tmp=new String(row);
                    json2.put("name", tmp);
                    json2.put("value",
Integer.parseInt(new String(a)));
                    ja.add(json2);
                }
                jaResult.put("tag_top", ja);
                System.out.println("---------------结束
tag_top表数据 END-----------------");
                System.out.println("---------------查询
job_info表数据 START-----------------");
                table =
connection.getTable(TableName.valueOf(job_info_table)
);
```

```
                ja=new JSONArray();
                resultx = table.getScanner(new Scan());
                for (Result result : resultx) {
                    byte[] row = result.getRow();
                    System.out.println("row key is:" +
new String(row));
                    byte[] a =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("city"));
                    byte[] b =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("company_name"));
                    byte[] c =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("tag"));
                    byte[] d =
result.getValue(Bytes.toBytes("info"),
Bytes.toBytes("salary"));
                    String str=new String(a);
                    String strc=new String(c);
                    String strb=new String(b);
                    String strd=new String(d);
```

```
                    if(!str.equals("未分类
")&&!strc.contains("\"")&&!strb.contains("\"")&&!strd
.contains("\"")&&!str.contains("\"")) {
                        JSONObject json2=new JSONObject();
                        json2.put("city", str);
                        json2.put("company_name", strb);
                        json2.put("tag",strc);
                        json2.put("salary", strd);
                        ja.add(json2);
                        }
                }
                System.out.println("---------------查询
job_info表数据 END-----------------");
                jaResult.put("job_info", ja);

            }catch(Exception e){
                e.printStackTrace();
            }
           return jaResult.toString();
       }
    }
```

2）Start 类

Start 类是服务器启动时就会运行的定时任务类。

```
public class Start implements ServletContextListener{
   public static String STR="";
   private static int I=0;
    @Override
       public void
contextInitialized(ServletContextEvent event) {
       I++;
       System.out.println("访问Hbase"+I);
       HbaseDemo h=new HbaseDemo();

//STR=h.queryAllTable("h_top_tag_gjc5","h_tag_salary_
top","h_city_tag_salary","h_tag_top","h_job_info2");

STR=h.queryAllTable("h_top_tag_gjc_spys","h_tag_salar
y_top_spys","h_city_tag_salary_spys","h_tag_top_spys"
,"h_job_info_spys");
       }
       @Override
       public void
```

```
contextDestroyed(ServletContextEvent event) {
        }
 }
```

3）打包使用

把所有代码（包括前端页面等等）打包成 war 文件，上传至 linux 中，并复制进到 tomcat 中的 webapps 文件夹内，比如：/usr/tomcat/webapps。再次进入到 tomcat 中的 bin 目录，运行 startup.sh 文件启动 tomcat。

4.shell 自动化

```
#!/bin/bash
#版本

function read_file(){
version="t3.1"

arr_data=('jobs1.csv' 'jobs2.csv' 'jobs3.csv' 'jobs4.csv' 'jobs5.csv' 'jobs6.csv')
arr_jar=('xie-mr1.jar' 'xie-udf2.jar')
arr_web=('BigDataTest.war')
arr_bs=('data' 'jar' 'web')
```

```
count=0


echo "程序开始"
echo "版本$version"
echo "正在检测文件是否缺失"
if [ -f "/usr/hadoop/shell/t3.sh" ]
        then
        count=` expr 1 + $count `
        else
        echo "/usr/hadoop/shell/t3.sh 未确认"
fi


for((i=0;i<${#arr_bs[@]};i++))


do
eval tmp=\${arr_${arr_bs[$i]}[@]}
for data in $tmp
do
        if [ ! -f "/usr/hadoop/${arr_bs[$i]}/$data" ]
                then
                echo "/usr/hadoop/${arr_bs[$i]}/$data 未确认"
                break 2
```

```
            else
            echo "$data 已确认"
        fi
        count=` expr 1 + $count `
done

done
if [ $count == 10 ]
then
        echo "文件检测完毕"
        upload
else
        echo "文件不完整，准备退出"
fi
}

function upload(){

        echo "准备上传数据"
        hadoop fs -rmr /data
        hadoop fs -rmr /jar
        hadoop fs -mkdir /data
```

```
hadoop fs -mkdir /jar

hadoop fs -put /usr/hadoop/data/* /data

hadoop fs -put /usr/hadoop/jar/* /jar


echo "上传完毕"

echo "准备进行初步清洗"

echo "该过程可能需要 4-8 分钟"

echo "具体情况视机器而定"


hadoop fs -test -e /out

if [ $? == 1 ]

then

    `hadoop fs -mkdir /out`

fi

for dir in ${arr_data[@]}

do

    echo "正在清洗$dir"

`hadoop fs -test -e /out/$dir`

if [ $? == 0 ]

then

    `hadoop fs -rmr /out/$dir`

fi
```

```
        `hadoop jar /usr/hadoop/jar/xie-mr1.jar test.mapreduce.t2.Main
/data/$dir /out/$dir`
        done

        echo "初步清洗完成"

        hive2
    }

    function hive2(){

        arr_table=("jobs_" "h_tag_top_" "h_top_tag_gjc_" "h_tag_salary_top_"
"h_city_tag_salary_" "h_job_info_")
        count=0

        echo "准备进行针对性清洗"
        echo "该过程可能需要 30-60 分钟"
        echo "具体情况视机器而定"

        tmp_path="/usr/hadoop/shell/tmp.txt"
        if [ -f $tmp_path ];then
            `rm -f $tmp_path`
```

```
        fi



        hive -e "drop function fenci"
        hive -e "drop function fenci_city"
        hive -e "drop function fenci_salary"
        hive -e "create function fenci as 'test.hive.udf.t2.fenciUDF' using jar 'hdfs:///jar/xie-udf3.jar'"
        hive -e "create function fenci_city as 'test.hive.udf.t1.CityUDF' using jar 'hdfs:///jar/xie-udf2.jar'"
        hive -e "create function fenci_salary as 'test.hive.udf.t1.SalaryUDF' using jar 'hdfs:///jar/xie-udf2.jar'"


    while ((1)); do
        count2=0

        for ((i=0;i<${#arr_table[@]};i++))
            do
            hadoop fs -test -e /user/hive/warehouse/${arr_table[$i]}$count
            if [ $? == 0 ]
                then
```

```
                count=` expr 1 + $count `
                break
            fi
            count2=` expr 1 + $count2 `
        done

        if [ $count2 == ${#arr_table[@]} ]
            then
            break
        fi
    done
    count2=$count
    count=0

    #第一张 jobs 表
    table_name=${arr_table[$count]}$count2
    echo "开始创建$table_name"
    hive -e "create table $table_name(company_financing_stage string,company_industry string,company_location string,company_name string,company_nature string,company_overview string,company_people string,job_edu_require string,job_exp_require string,job_info string,job_name string,job_salary string,job_tag string,job_welfare string) row format delimited
```

```
fields terminated by '\u0001';"


    for dir in ${arr_data[@]}
    do
        for (( j = 0; j < 10; j++ )); do
        `hadoop fs -test -e /out/$dir/part-r-0000$j`
        if [ $? == 0 ]
        then
            `hive -e "load data inpath '/out/$dir/part-r-0000$j' into table $table_name;"`
        else
            break
        fi
        done
    done


    hive -e "SELECT COUNT(job_name) FROM $table_name"
    echo "$table_name 清洗完毕"



    #第二张 h_tag_top 表
    count=` expr 1 + $count `
```

```
table_name=${arr_table[$count]}$count2
echo $table_name >> $tmp_path
echo "开始创建$table_name"
hive -e "create table $table_name
(job_tag string,count int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,info:count')
TBLPROPERTIES ('hbase.table.name' = '$table_name',
'hbase.mapred.output.outputtable' = '$table_name');"
hive -e "insert into $table_name
select job_tag,count(1) as count from ${arr_table[0]}$count2 group by job_tag order by count desc limit 10;"

hive -e "select * from $table_name"
echo "$table_name 清洗完毕"

#第三张 h_top_tag_gjc 表
count=` expr 1 + $count `
table_name=${arr_table[$count]}$count2
echo $table_name >> $tmp_path
echo "开始创建$table_name"
hive -e "create table $table_name
```

```
(job_tag string,gjc string,count string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,info:job_info,info:count')
TBLPROPERTIES ('hbase.table.name' = '$table_name',
'hbase.mapred.output.outputtable' = '$table_name');"

hive -e "insert into $table_name
select a.job_tag,concat_ws(',',collect_set(a.gjc)) as gjc_z,concat_ws(',',collect_set(cast(a.count as string))) as count_z from
(select gjc,job_tag,count(1) as count from (select job_tag,gjc from ${arr_table[0]}$count2
lateral view explode(split(fenci(job_info),','))t1 as gjc)b,${arr_table[1]}$count2 where b.job_tag=${arr_table[1]}$count2.job_tag group by b.gjc,b.job_tag)a
group by a.job_tag;"

echo "$table_name 清洗完毕"

#第四张职业均薪 TOP
count=` expr 1 + $count `
table_name=${arr_table[$count]}$count2
echo $table_name >> $tmp_path
```

```
echo "开始创建$table_name"

hive -e "create table $table_name
(job_tag string,salary int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,info:salary')
TBLPROPERTIES ('hbase.table.name' = '$table_name',
'hbase.mapred.output.outputtable' = '$table_name');"
hive -e "insert into $table_name
select tag,cast(avg(salary) as int) as salary from
(select job_tag as tag,fenci_salary(job_salary) as salary from
${arr_table[0]}$count2)a
where salary!=0
group by tag
order by salary desc
limit 10;"

echo "$table_name 清洗完毕"

#第五张城市招聘信息
count=` expr 1 + $count `
table_name=${arr_table[$count]}$count2
```

```
echo $table_name >> $tmp_path
echo "开始创建$table_name"

hive -e "create table $table_name
(city string,tag string,count string,salary int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,info:tag,info:count,info:salary')
TBLPROPERTIES ('hbase.table.name' = '$table_name',
'hbase.mapred.output.outputtable' = '$table_name');"
hive -e "insert into $table_name
select location,concat_ws(',',collect_set(job_tag)),concat_ws(',',collect_set(cast(count as string))),cast(avg(salary2) as int) from
(select fenci_city(company_location) as location,job_tag,count(1) as count,avg(salary) as salary2 from
(select company_location,job_tag,fenci_salary(job_salary) as salary from ${arr_table[0]}$count2)b
where salary!=0
group by company_location,job_tag)a
group by location;"
```

```
echo "$table_name 清洗完毕"

#第六张招聘信息表
count=` expr 1 + $count `
table_name=${arr_table[$count]}$count2
echo $table_name >> $tmp_path
echo "开始创建$table_name"

hive -e "create table $table_name
(id int,city string,company_name string,job_name string,salary string,job_tag string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,info:city,info:company_name,info:job_name,info:salary,info:job_tag')
TBLPROPERTIES ('hbase.table.name' = '$table_name',
'hbase.mapred.output.outputtable' = '$table_name');"

hive -e "insert into $table_name
select row_number() over (order by job_tag) as id,z.* from
(
select distinct * from(
```

```
select location,company_name,job_name,job_salary,job_tag from
(select fenci_city(${arr_table[0]}$count2.company_location) as location,${arr_table[0]}$count2.company_name,${arr_table[0]}$count2.job_name,${arr_table[0]}$count2.job_salary,${arr_table[0]}$count2.job_tag
from ${arr_table[0]}$count2,${arr_table[1]}$count2 where ${arr_table[0]}$count2.job_tag=${arr_table[1]}$count2.job_tag limit 700)a
union all
select location,company_name,job_name,job_salary,job_tag from
(select * from(select fenci_city(${arr_table[0]}$count2.company_location) as location,${arr_table[0]}$count2.company_name,${arr_table[0]}$count2.job_name,${arr_table[0]}$count2.job_salary,${arr_table[0]}$count2.job_tag
from ${arr_table[0]}$count2)bb,${arr_table[4]}$count2 where bb.location=${arr_table[4]}$count2.city limit 1300)b
)c
)z;
"

echo "$table_name 清洗完毕"

echo "hive 清洗完毕"

tomcat
```

```
}

function tomcat(){
tomcat_path="/usr/tomcat"
web_dir_name='BigDataTest'
echo "tomcat 开始部署"

if [ -f $tomcat_path/webapps/${arr_web[0]} ];then
`rm -f "$tomcat_path/webapps/${arr_web[0]}"`
fi

cp /usr/hadoop/${arr_bs[2]}/${arr_web[0]} $tomcat_path/webapps/${arr_web[0]}

if [ -d $tomcat_path/webapps/$web_dir_name ];then
`rm -rf "$tomcat_path/webapps/$web_dir_name"`
fi

startup.sh

cp $tmp_path $tomcat_path/webapps/$web_dir_name/WEB-INF/classes/other/table.txt
```

```
echo "tomcat 部署完毕"
echo "程序结束"
}
read_file
#hive2
```