基于大数据平台的舆情热词分析

# 背景介绍

## 大数据平台



图1.1 Hadoop大数据平台结构

大数据平台是指为了处理高速涌现的大量多样化数据（大数据），使用分布式计算框架，建立计算集群，并在上面运行各种计算任务而构建的计算平台。

## 舆情分析

舆情是大众意见观点信息合集的总称，所以舆情工作主要是监测网上的各种舆情舆论信息，如媒体的议题、观点，网民的意见、建议、褒贬与评价等。

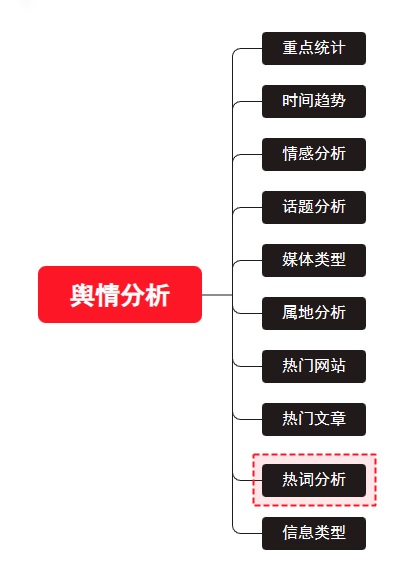


图1.2 舆情分析的内容

舆情分析的主要内容包括重点统计、时间趋势、情感分析、话题分析、媒体类型、属地分析、热门网站、热门文章、热词分析、信息类型等等。

## 基于大数据平台的舆情热词分析

互联网+时代的来临，移动互联网技术的快速发展，并且在各个领域中应用，为其他行业的发展提供了技术支持。随时随地发布新闻、了解咨询、关注国计民生以及发表个人观点和看法成为新常态。

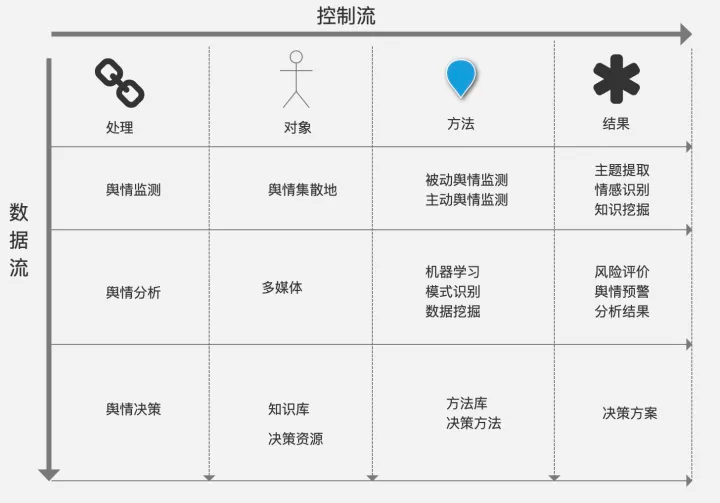


图1.3 基于大数据的舆情分析与监控流程

舆情的发生、发展、演化及传播等特点发生着翻天覆地的变化，与之相应的舆情监测、分析和决策方法日益成为公司部门关注的焦点。利用信息技术、舆情监测等方面理念、理论及方法对网络舆情的演化发展进行了大量研究，提出面向大数据的网络舆情监测：发现舆情主题，分析情感倾向，设计主题，传播趋势；采用数据挖掘技术在事前、事中和事后分三个阶段对舆情进行分析，对其风险进行评价，预测其发展趋势，及时提出预警。



图1.4 舆情热词分析词云

大数据的出现和大数据平台的发展，为舆情的收集、分析和展示开拓了新空间、赋予了新的能力。舆情分析工作中，热词分析工作成为舆情分析与大数据结合的代表性工作，而高频词云分析成为大数据技术赋能热词分析的代表性工作。因此我组拟构建一个基于大数据平台的舆情热词分析项目，通过大数据平台搭建、数据采集、数据预处理、大数据算法开发、前端展示等工作探究大数据平台在舆情分析工作中的应用。

# 架构设计

数据采集

数据处理

HDFS

数据存储

开源数据集

数据清洗/分词

计算

分析与展示

MapReduce

Hive+词云

图2.1 架构设计

本项目的架构设计如上图所示，左边为逻辑架构，右边为对应的技术方法，基于大数据平台的舆情热词分析项目的逻辑流程为：

1. 数据采集，采集开源数据集的中文新闻语料数据作为大数据分析的基础数据
2. 数据处理，实现数据采集端的数据预处理流程，将原始语料进行数据清洗并进行中文分词
3. 数据储存，将预处理的数据上传至HDFS，用于大数据计算
4. 计算，设计算法使用Hadoop-Mapreduce计算架构完成词频统计和排序工作
5. 分析与展示，将词频统计结果导入到Hive中，并设计前端页面进行展示。

# 任务分工

|  |  |
| --- | --- |
| 任务 | 负责同学 |
| 架构设计 | 陈子豪 李龙 孙辉泽 |
| 环境搭建 | 孙辉泽 |
| 数据采集 | 陈子豪 |
| 数据处理 | 陈子豪 李龙 孙辉泽 |
| 算法开发 | 李龙 |

表1.1 任务分工表

本项目小组成员共三人，任务分工如上表所示，架构设计和数据处理由三人共同完成，环境搭建、数据采集和算法开发工作每个同学负责一个板块。

# 环境搭建

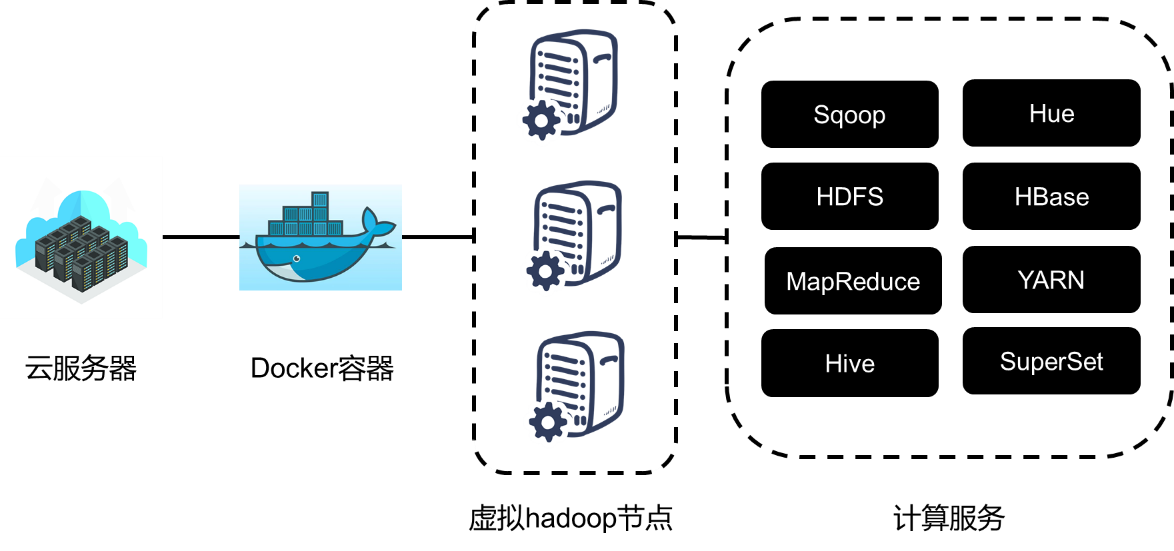


图4.1 基于Docker的Hadoop集群概念图

基于Docker部署的Hadoop计算集群如上图所示，使用一台云服务器作为物理计算资源，在云服务器上安装Docker容器，使用Dokcer容器部署虚拟Hadoop节点，并组成Hadoop计算集群，在集群之上提供HDFS、YARN、HIVE等计算服务。

## 云服务器环境



图4.2 云服务器配置

我们使用了一台华为云HECS服务器作为小组的协作平台，具体配置如下：

云服务器名称：华为云HECS

硬件配置：2核 4GB内存 2Mb带宽

操作系统：CentOS 7.9 64bit

## Docker环境

### Docker简介

容器技术、虚拟化技术已经成为一种被大家广泛认可的服务器资源共享方式，容器技术可以在按需构建操作系统实例的过程当中为系统管理员提供极大的灵活性。由于hypervisor虚拟化技术仍然存在一些性能和资源使用效率方面的问题，因此容器技术（Container）结合虚拟化技术的解决方案被业界广泛使用。

容器的集装箱概念是参照了航运中的集装箱概念，通过集装箱将运输的流程规范成了标准化的操作，Docker容器具备自包含的能力，将自身程序所依赖的程序全部包含在了容器中，通过Docker将底层环境打平，用户可以将一个容器镜像运行在任何操作系统的宿主机上，也就是Docker所说的“Build once, Run anywhere”。

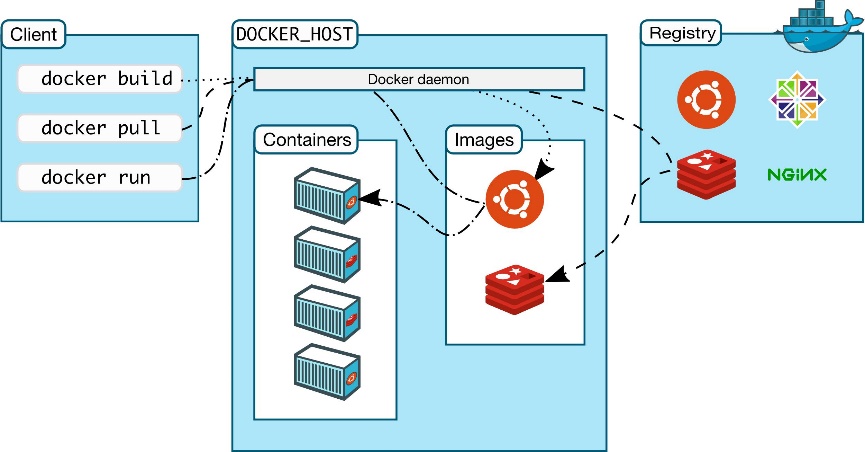


图4.3 Docker技术原理

Docker的技术原理如上图所示，通过Docker Client将需要执行的Docker命令发送给Docker运行的节点上的Docker daemon，Docker daemon将我们的请求进行分解执行，例如我们执行Docker build命令它会根据Dockerfile构建一个镜像存放于本地，执行Docker pull命令会从远端的容器镜像仓库拉取镜像到本地，执行Docker run命令会将容器镜像拉取并运行成为容器实例。

使用Docker容器技术，可以允许我们在同一台服务器上创建多个Hadoop节点的虚拟镜像，每个镜像环境都是相对独立的，模拟了分布式的环境。因此，利用Docker技术可以在一个物理计算机上搭建出一整个Hadoop云计算集群，便于小组协作。

### Docker安装

在配置好Docker运行的环境后，执行以下命令安装Docker：

[root@hecs-161794 ~]# yum install docker-ce

[root@hecs-161794 ~]# systemctl start docker

[root@hecs-161794 ~]# systemctl enable docker

Docker安装好后，输入命令

[root@hecs-161794 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

可以看到Docker已正常运行。

## Hadoop环境

### Hadoop简介

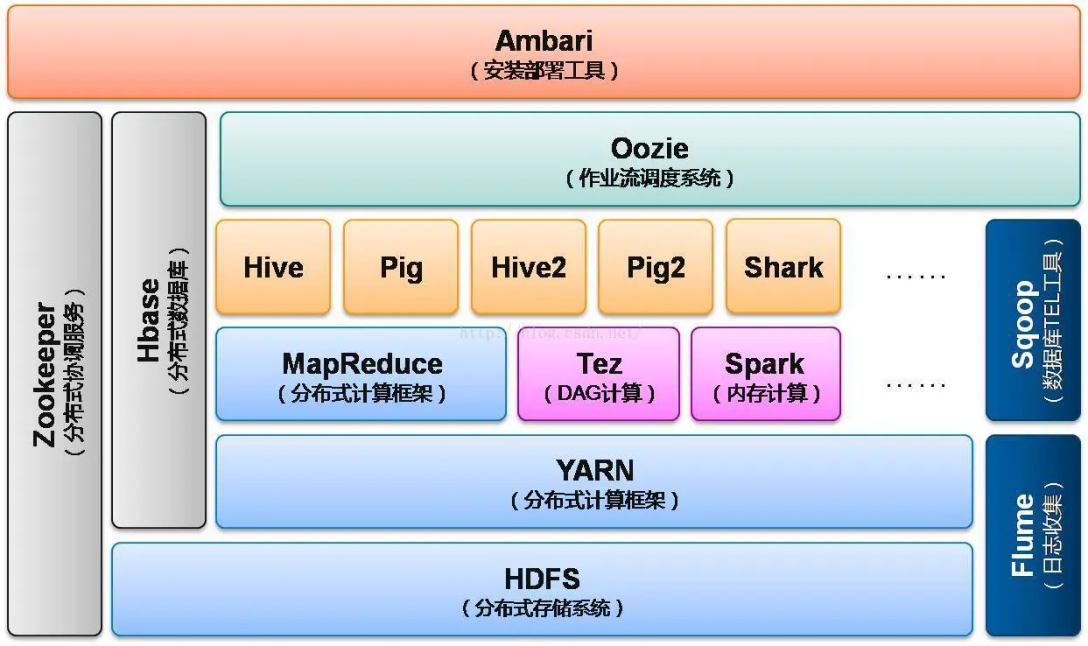


图4.4 Hadoop架构

Hadoop是目前应用广泛的分布式计算平台，拥有繁荣的生态圈，Hadoop包含的组件结构如上图所示，包括安装部署工具Ambari、分布式协调服务Zookeeper、分布式数据库Hbase、数据仓库Hive、分布式计算框架MapReduce、内存计算Spark、分布式计算框架YARN、分布式存储系统HDFS、数据库TEL工具Sqoop等等。我组气象分析应用中主要使用的框架包括HDFS、YARN、Hive和Sqoop，其中最为关键的是HDFS和YARN的架构。

1. **HDFS**

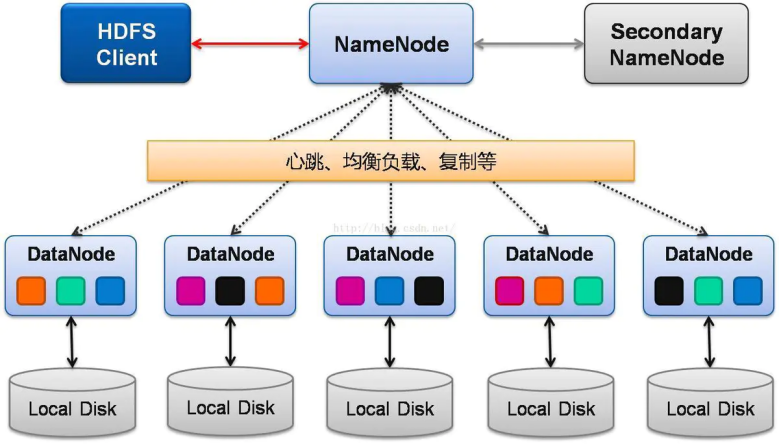


图4.6 HDFS架构

HDFS是Hadoop的文件存储系统，HDFS采用的是主从架构，NameNode是集群中的主节点，用于管理集群中的各种数据。DataNode是集群中的从节点，用于分布式储存集群中的各种数据。

1. **YARN**

YARN是Hadoop2.x之后使用的的分布式计算架构。区别于Hadoop1.x的Mapreduce架构，YARN使用了ResourceManager和NodeManager而非Mapreduce的Jobtracker和TaskTracker，使计算任务调度工作也可由集群的其他节点完成，使任务调度不再成为瓶颈。

YARN架构如下图所示，其中角色包括ResourceManager和NodeManager。ResourceManager负责接收计算请求任务，和集群的资源分配。NodeManager负责作为ApplicationMaster调度任务，同时负责任务计算。

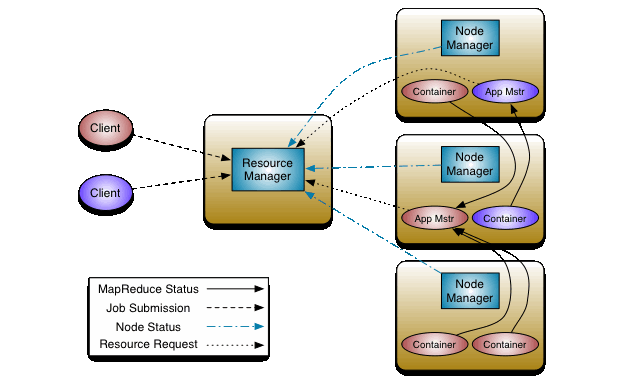


图4.7 YARN架构

### Docker-Compose简介

Docker Compose是一个用来定义和运行复杂应用的Docker工具。一个使用Docker容器的应用，通常由多个容器组成。使用Docker Compose不再需要使用shell脚本来启动容器。

Compose 通过一个配置文件来管理多个Docker容器，在配置文件中，所有的容器通过services来定义，然后使用docker-compose脚本来启动，停止和重启应用，和应用中的服务以及所有依赖服务的容器，非常适合组合使用多个容器进行开发的场景。因此使用Docker-Compose可以简化Hadoop的部署流程，更快完成项目中计算平台的上线。



图4.8 用于配置Hadoop集群的docker-compose文件

Docker-Compose的配置文件docker-compose.yml描述了每个Hadoop节点及其依赖和属性，hadoop.env文件描述了基于Dockerd的Hadoop集群需要的基础环境。

### Hadoop配置与部署

拟构建的Hadoop集群结构如下图，集群全部部署在Docker容器上，包括：一个NameNode、三个DataNode、一个NodeManager、一个ResourceManager和一个HistoryManager。 其中NameNode作为外部访问的端口，Docker将主机的9870端口与NameNode的9870端口完成外部映射，云服务器安全组打开9870端口，即可从外部访问Hadoop集群。DataNode用于HDFS数据存储。NodeManager和ResourceManger用于大数据计算，HistoryManager用于管理历史日志。

Docker

NameNode

DataNode1

DataNode2

DataNode3

ResourceManager

NodeManager

HistoryManager

图4.9 Hadoop集群结构

Docker-compose.yml文件中描述了Hadoop各节点的属性，包括使用的Docker镜像、磁盘映射和依赖项，具体的配置文件如下：

[root@hecs-161794 docker-hadoop]# vim docker-compose.yml

version: "3"

services:

namenode:

image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8

container\_name: namenode

restart: always

ports:

- 9870:9870

- 9000:9000

volumes:

- hadoop\_namenode:/hadoop/dfs/name

environment:

- CLUSTER\_NAME=test

env\_file:

- ./hadoop.env

datanode1:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container\_name: datanode1

restart: always

volumes:

- hadoop\_datanode1:/hadoop/dfs/data

environment:

SERVICE\_PRECONDITION: "namenode:9870"

env\_file:

- ./hadoop.env

datanode2:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container\_name: datanode2

restart: always

volumes:

- hadoop\_datanode2:/hadoop/dfs/data

environment:

SERVICE\_PRECONDITION: "namenode:9870"

env\_file:

- ./hadoop.env

datanode3:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container\_name: datanode3

restart: always

volumes:

- hadoop\_datanode3:/hadoop/dfs/data

environment:

SERVICE\_PRECONDITION: "namenode:9870"

env\_file:

- ./hadoop.env

resourcemanager:

image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8

container\_name: resourcemanager

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 datanode2:9864 datanode3:9864"

env\_file:

- ./hadoop.env

nodemanager1:

image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8

container\_name: nodemanager

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 datanode2:9864 datanode3:9864 resourcemanager:8088"

env\_file:

- ./hadoop.env

historyserver:

image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8

container\_name: historyserver

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 datanode2:9864 datanode3:9864 resourcemanager:8088"

volumes:

- hadoop\_historyserver:/hadoop/yarn/timeline

env\_file:

- ./hadoop.env

volumes:

hadoop\_namenode:

hadoop\_datanode1:

hadoop\_datanode2:

hadoop\_datanode3:

hadoop\_historyserver:

Hadoop集群共分配了6个节点，包括：namenode（管理命名空间）、datanode1、datanode2（提供读写请求）、nodemanager（运行程序并监控节点）、resourcemanager（全局资源监控）、historyserver（记录作业信息）节点各一个。

通过`volumes`，可以将hdfs的文件映射到本地，这样一来即使我们删除了容器，只要compose目录下的文件都在，那么我们在创建了新的集群后，原有的hdfs文件就会恢复，不需要重新上传。

使用Docker-Compose，Hadoop集群的启动只需要输入命令：

[root@hecs-161794 docker-hadoop]# docker-compose up -d

Hadoop相关的Docker镜像即可开始运行，输入Docker查询命令可以看到7个节点都处于健康状态：

[root@hecs-161794 docker-hadoop]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e0bedcfb93af bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 8088/tcp resourcemanager

27ba7059e920 bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9870->9870/tcp, :::9870->9870/tcp namenode

cbb798ccfe75 bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 8042/tcp nodemanager

435a231fe350 bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 8188/tcp historyserver

360b3450da85 bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 9864/tcp datanode1

e7ab07fc33dd bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 32 seconds (healthy) 9864/tcp datanode3

75d82dcef521 bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run…" 33 seconds ago Up 31 seconds (healthy) 9864/tcp

浏览器访问服务器地址： <http://123.249.6.46:9870/> 即可进入Hadoop的NameNode主界面，也可以访问DataNode管理界面和HDFS管理界面。

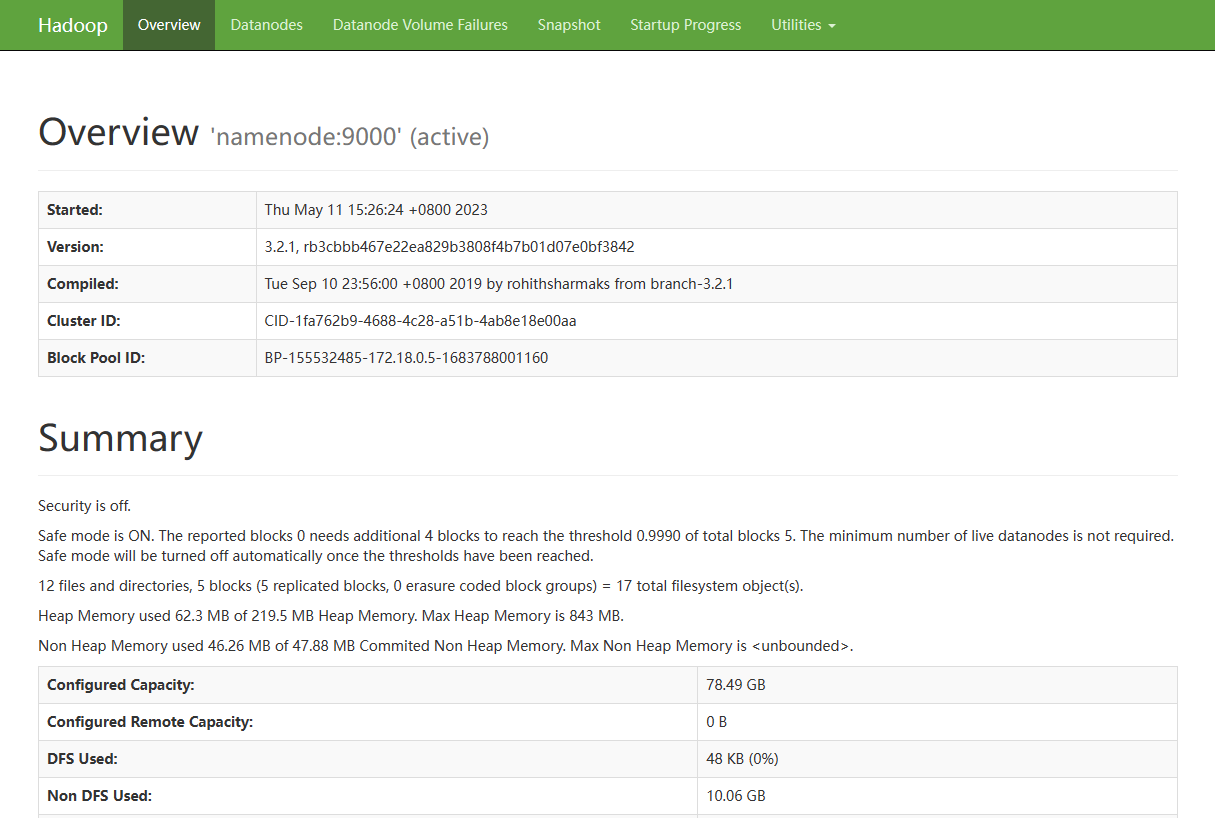


图4.10 Hadoop界面

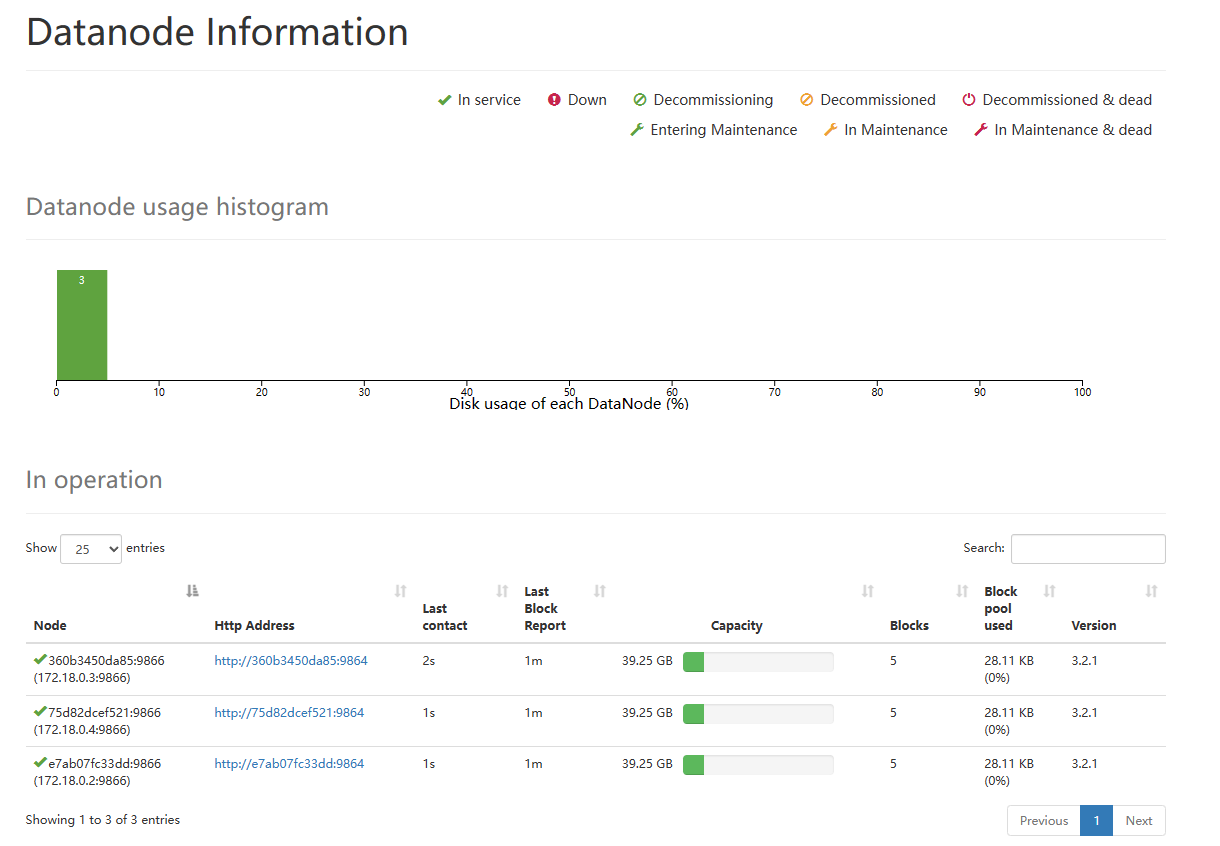


图4.11 DataNode管理界面

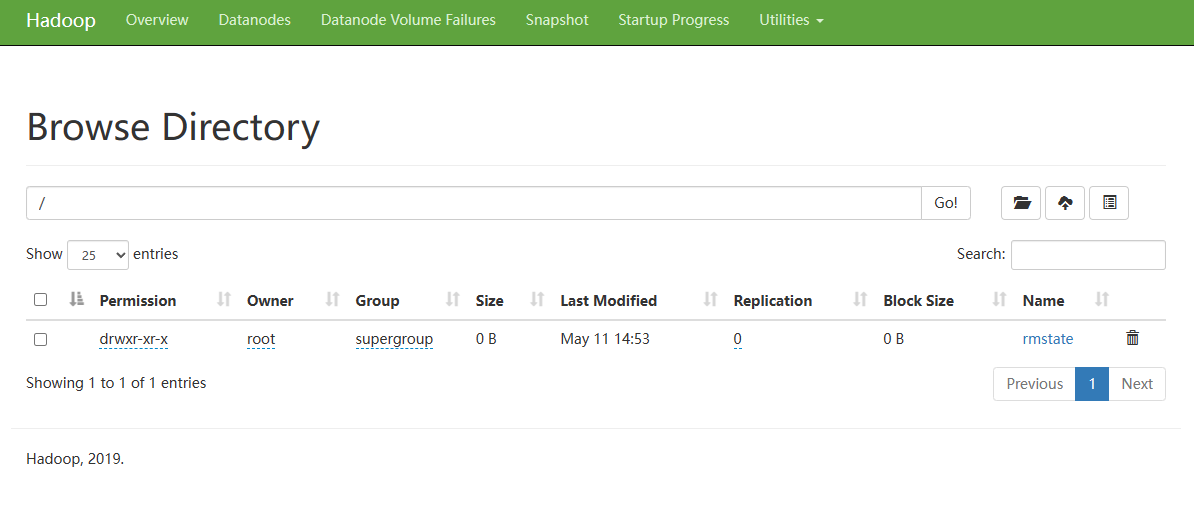


图4.12 HDFS管理界面

### Hive环境

Hive是基于[Hadoop](https://baike.baidu.com/item/Hadoop/3526507?fromModule=lemma_inlink)的一个[数据仓库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%BB%93%E5%BA%93/381916?fromModule=lemma_inlink)工具，用来进行数据提取、转化、加载，这是一种可以存储、查询和分析存储在Hadoop中的大规模数据的机制。Hive数据仓库工具能将结构化的数据文件映射为一张数据库表，并提供[SQL](https://baike.baidu.com/item/SQL/86007?fromModule=lemma_inlink)查询功能，能将[SQL语句](https://baike.baidu.com/item/SQL%E8%AF%AD%E5%8F%A5/5714895?fromModule=lemma_inlink)转变成[MapReduce](https://baike.baidu.com/item/MapReduce/133425?fromModule=lemma_inlink)任务来执行

Hive的搭建主要分为两个部分，分别为Hive在Hadoop中的配置、Hive元数据的配置：

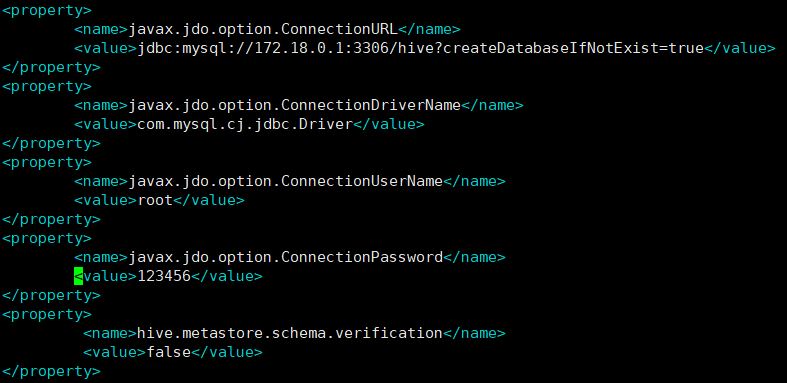
1. Hive在Hadoop中的配置  
    主要就是解决Hive、Hadoop中jar包的冲突问题
2. Hive元数据的配置  
    Hive可以选择多种数据库作为元数据管理后端，这里选择MySQL。相关配置 主要包括修改hive-site.xml、安装MySQL驱动，使得Hive可以连接到指定数据库。  
    

图4.13 Hive配置

# 数据采集

# 数据处理

## 数据清洗

选择数据集中的训练数据作为本项目的语料，首先进行数据清洗，为了方便管理，按每一类整理为一个txt文件。

处理前的语料如下：

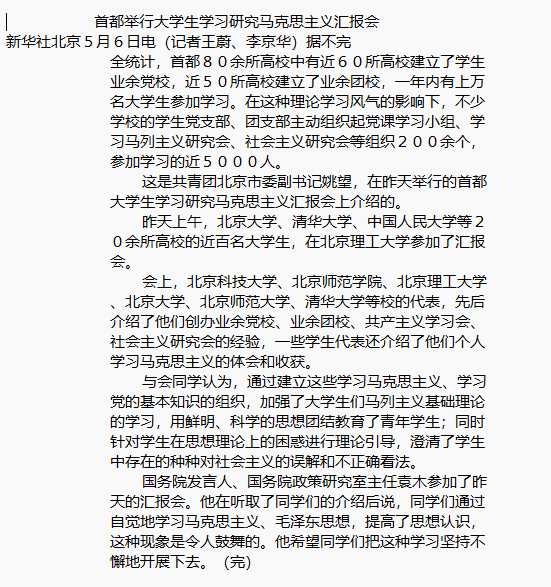


图6.1原始数据示例

数据清洗去掉语料中的所有空格、制表符、回车、数字和字母，只保留汉字和标点。

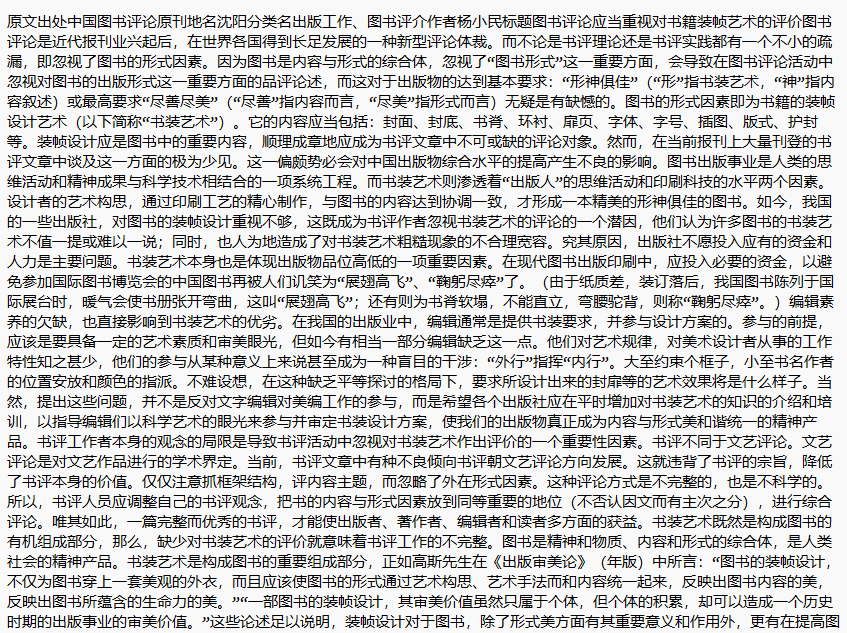


图6.2 清洗后的数据示例

## 线下分词

分词任务根据任务属性、数据量的大小可以分为在线分词和线下分词两种方式。线下分词是指统一获取的数据在本地完成分词后上传至大数据平台，这种模式适合本项目的静态开源数据集。而在线分词是在原始数据上传至大数据平台后，通过大数据计算的方式进行分词。

使用python进行分词，调用python的jieba中文分词库，将连续的语料划分为词语，并根据停用词库，过滤一些常用的停用词，使分词得到的大部分词语有明确含义。

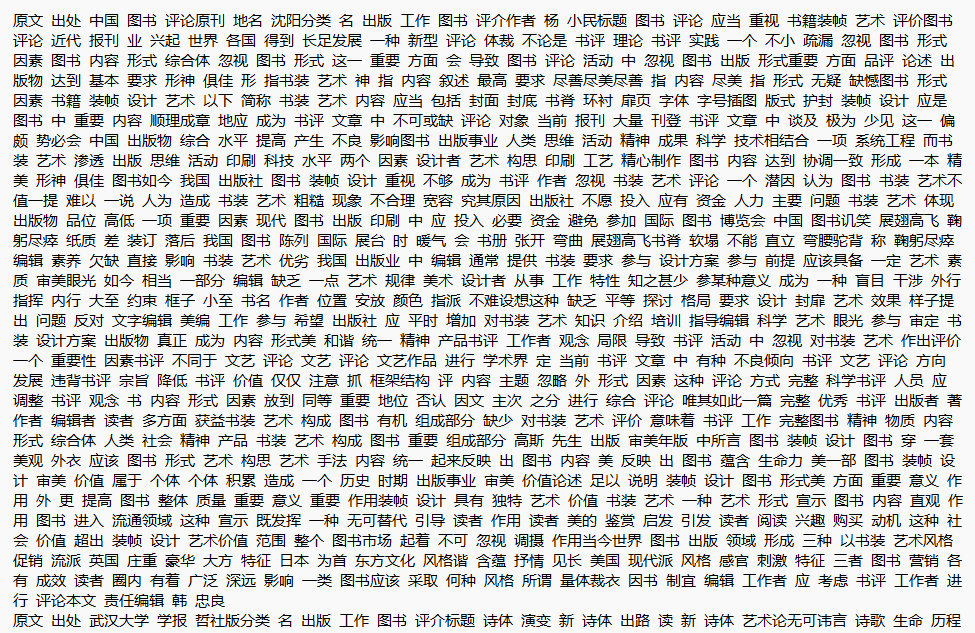


图6.3 分词后的数据示例

将分词结果上传至HDFS用于后续的词频统计。

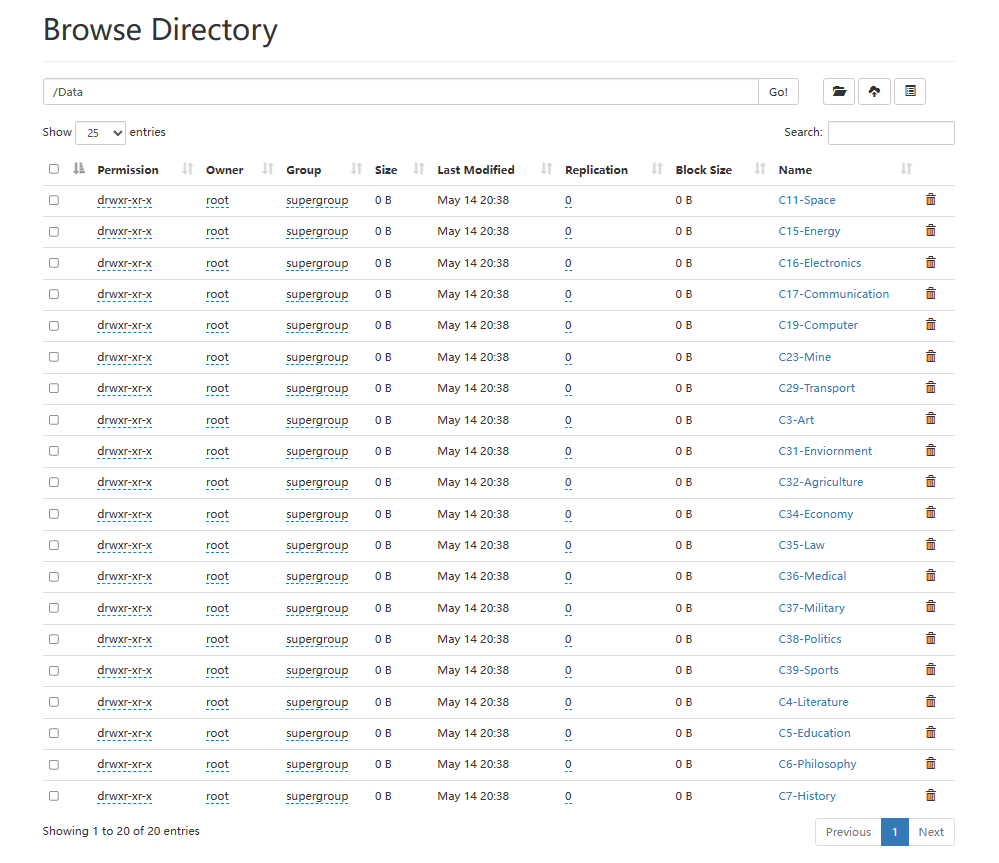


图6.4 分词上传结果

## 在线分词

在线分词使用Hadoop-streaming+python完成，Hadoop-streaming是hadoop提供的大数据计算包，可以让任何语言编写的map, reduce程序能够在hadoop集群上运行；map/reduce程序只要遵循从标准输入stdin读，写出到标准输出stdout即可。因此，将原始数据上传至HDFS，同样可以完成分词操作。

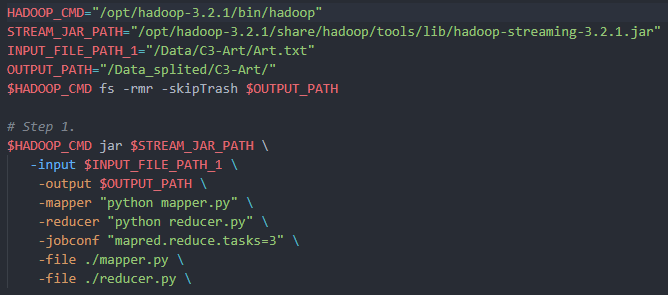


图6.5分词命令行脚本

# 算法开发

## WordCount

基于WordCount任务对分词结果进行处理。结果文件为一个txt文件，通过类名区分该分词结果所在的新闻类别。

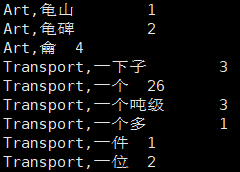


图7.1 分词结果

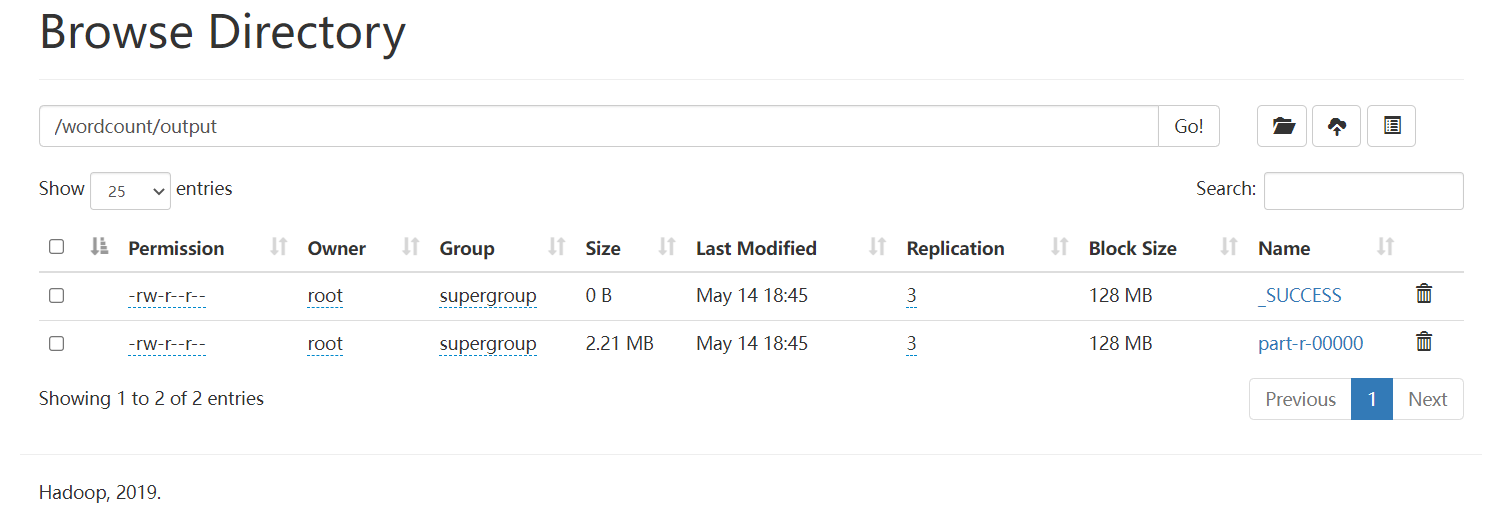


图7.2 分词结果上传

## 结果划分

针对WordCount产生的结果文件part-r-00000，利用MapperReduce根据每一行的类别标志将各个类别的结果分别提取到不同的文件。

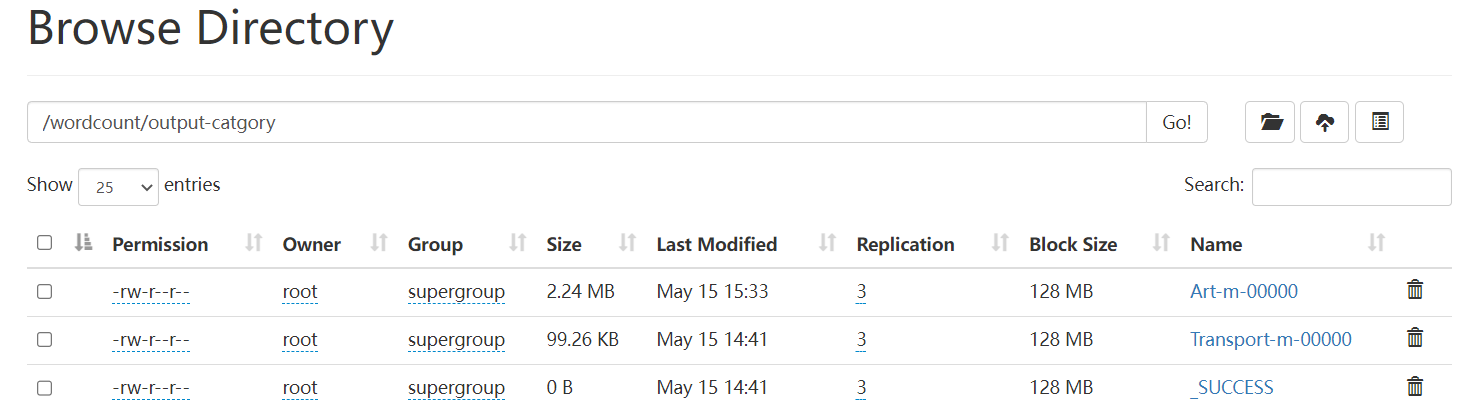


图7.3 结果划分

## 结果存储

将上述处理结果存储到Hive当中，以便后续处理，例如词云展示。

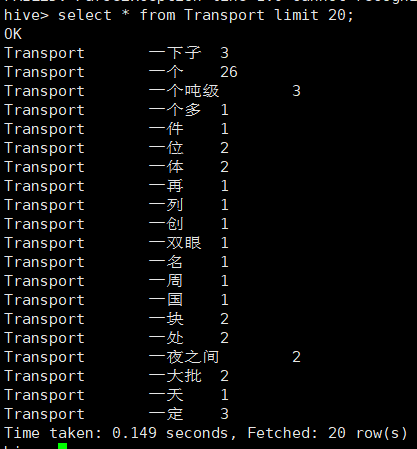


图7.4 Hive数据存储

# 结果展示