Hadoop wordcount demo

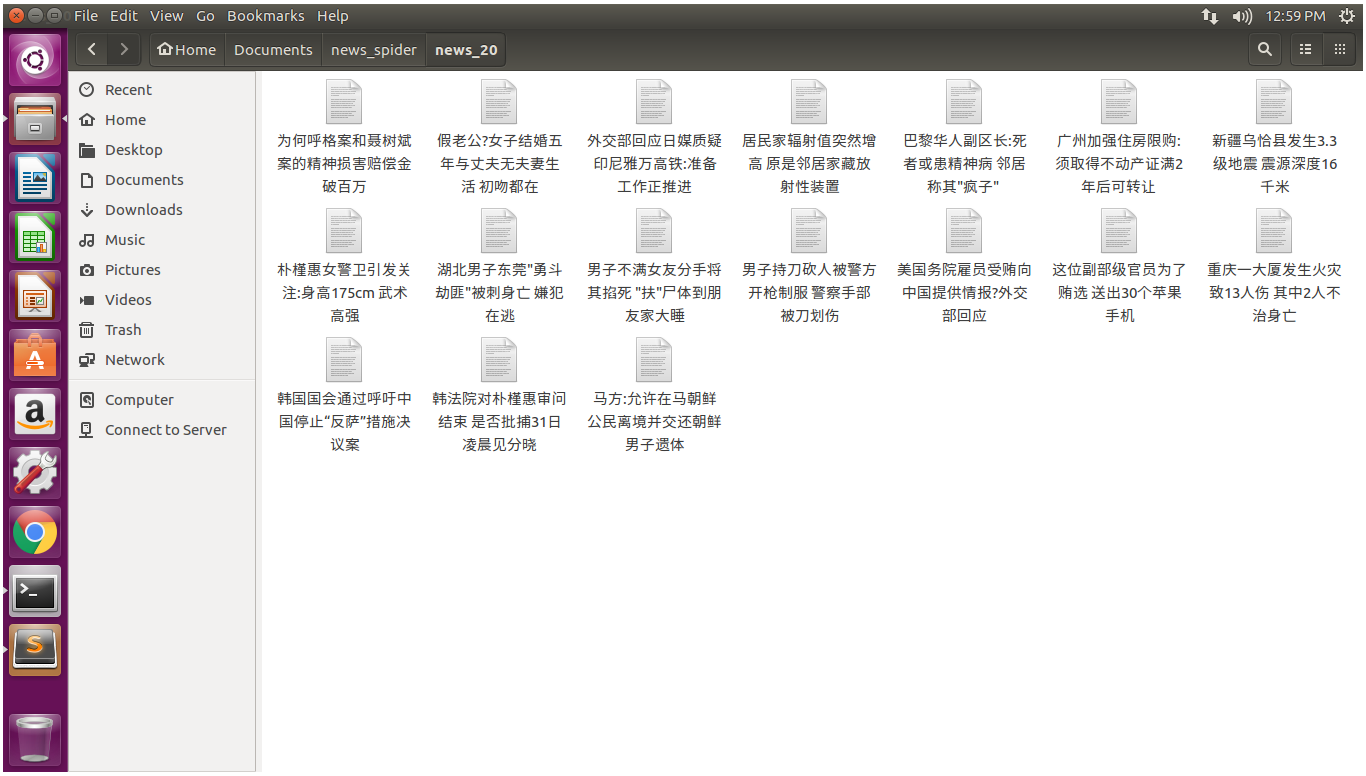
小组成员

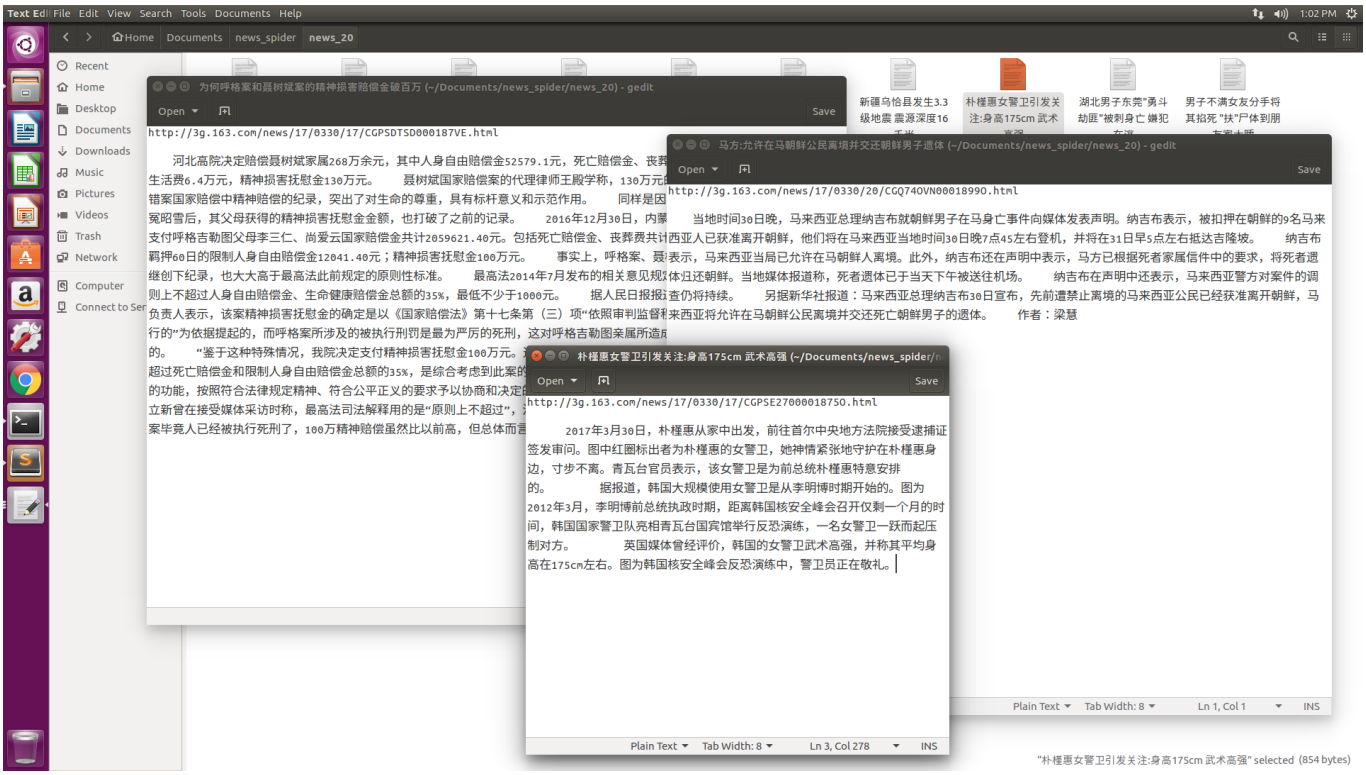
- 张俊杰 stuID: 51164500265
- 冯琰 stuID: 51164500207
- 李文浩 stuID: 51164500105

关于数据爬取，爬取了20条网易新闻，采用了python scrapy框架，核心代码如下：

```
# -*- coding: utf-8 -*-
import scrapy
import re
from news_spider.items import News

class NeteaseSpiderSpider(scrapy.Spider):
    name = "netease_spider"
    PAGE_SIZE = 25
    # start_urls = ['http://3g.163.com/touch/article/list/BBM54PGAwangning/0-5000.html']
    start_urls = ['http://3g.163.com/touch/article/list/BBM54PGAwangning/0-20.html']
    header = {"User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36\
     (KHTML, like Gecko) Ubuntu Chromium/56.0.2924.76 Chrome/56.0.2924.76 Safari/537.36"}
    news_count = 0

    def start_requests(self):
        yield scrapy.Request(url=self.start_urls[0], headers=self.header, callback=self.parse)

    def parse(self, response):
        content = response.xpath('//body/p/text()').extract()[0]
        # with io.open('news.txt', 'w') as f:
        #     f.write(content)
        urls = re.findall(
            'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',
            content)

        for url in urls:
            if url.endswith('html'):
                yield scrapy.Request(url=url, callback=self.parse_news)


    def parse_news(self, response):
        item = News()
        item['url'] = response.url
        item['title'] = response.xpath('/html/body/h1/text()').extract()[0]
        content = ''
        ps = response.xpath('//body/div[@class="content"]/p')
        for p in ps:
            content += p.xpath('./text()').extract()[0]
        item['content'] = content


        return item
```

爬取之后的数据：



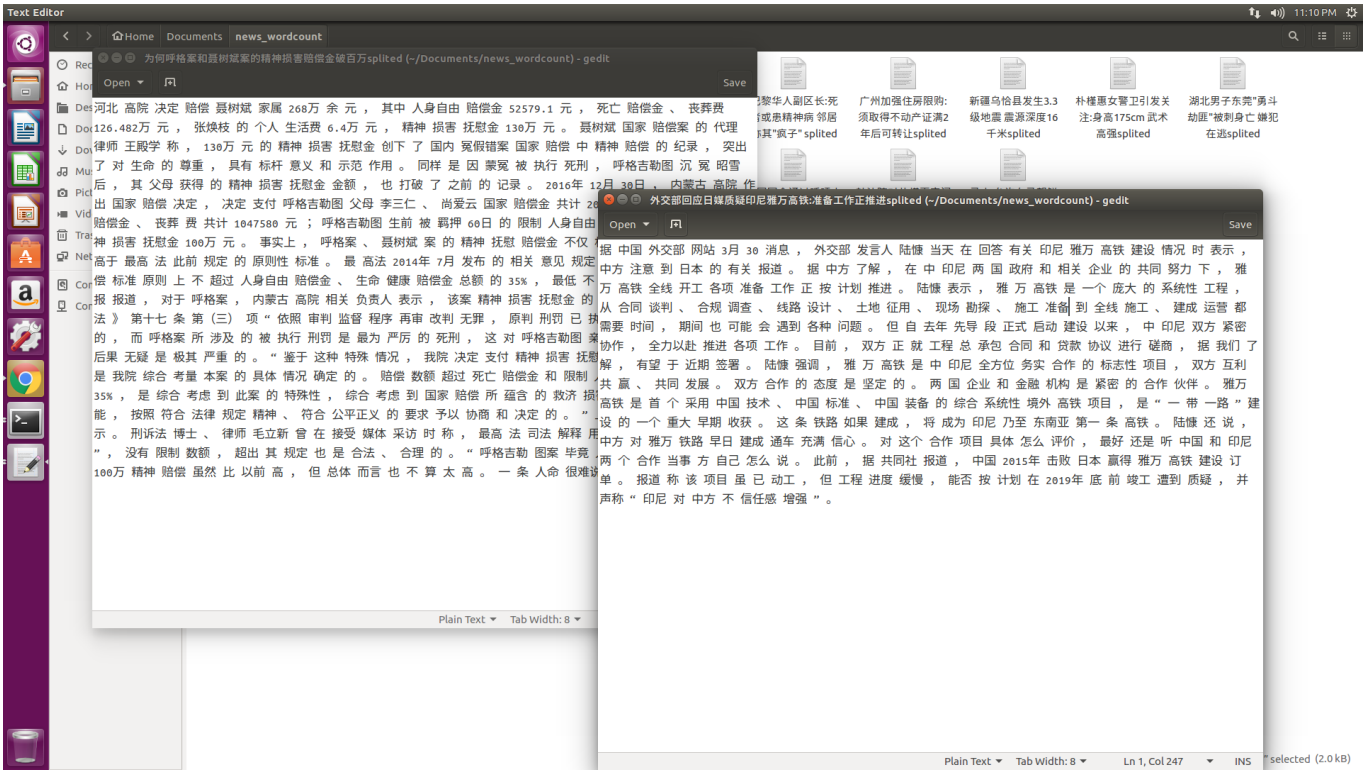


对应的URL存在爬取下来的文件当中，可以直接访问进行验证。

在进行wordcount的时候，如果源语言是英文，那么可以直接通过空格来进行word的拆分。但是对于中文还要进行分词，由于分词是一项比较大的工程，这里就直接使用了python的bosonnlp package，分词阶段的核心代码如下：

```python
import os
import io
from bosonnlp import BosonNLP

# Get news file path
news_dir = '/home/ewan/Documents/news_spider/news_20/'
file_list = os.listdir(news_dir)

# Get the news dict
news = {}
for index, file in enumerate(file_list):
    with io.open(news_dir + file, 'r') as f:
        news[index] = file.decode('utf-8') + '\n' + f.read()

nlp = BosonNLP('FuHSE7Vf.13924.jadflTdrQLWx')
splitted_news = {}

SIZE = len(news)

for key, value in news.items():
    result = nlp.tag([value])
    words = ''
    for index, word in enumerate(result[0]['word']):
            words += word + ' '
    splitted_news[key] = words

for index, content in splitted_news.items():
    with io.open(str(index), 'w') as f:
        f.write(content)
```

最后得到分词后的文件



下面贴出mapreduce的java代码：

```java
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

```

```
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Pig Latin：

```
A = load '/tmp/demo.txt';
B = foreach A generate flatten(TOKENIZE((chararray)$0)) as word;
C = filter B by word matches '\\w+';
D = group C by word;
E = foreach D generate COUNT(C), group;
store E into '/tmp/wordcount_output';
```

word count结果

```
"   6
(   2
)   2
1   6
1000    1
100万    3
1047580 1
10分 1
10日 1
10时 2
10月 1
10点 1
11  1
110万    1
11时 1
11点 1
12041.40    1
126.482万    1
12月 2
13  2
130万    2
15.6万   1
16千 2
175 2
188 1
189 1
19  1
1979年   1
19时 1
19点10分  1
1月  1
2   7
20  1
2011年   1
2012年   4
2013年   3
2014年   1
2015年   2
2016年   3
2017    1
2017年   2
2019年   1
2059621.40  1
207 1
21日 1
23日 2
25日 2
26  2
260万    1
268万    1
26日 1
27  2
27日 1
28日 2
29日 2
```

```
3   3
3.3 2
30  5
300 1
30分 1
30日 14
31日 4
35  2
35% 2
38  1
39.91   1
3月  12
4   2
40% 1
42  1
45  1
4S  3
4日  1
4月  2
5   3
5196亿   1
52579.1 1
5点  1
6.4万    1
60日 1
65  1
6日  2
75.13   1
76  1
7分  1
7日  1
7月  2
7点  2
7点43分   1
8   1
850 1
9   5
9月  1
:   5
;   2
?   2
APP 1
X   3
XQ2505  2
cm  2
http://3g.163.com/news/17/0330/17/CGPSDTSD000187VE.html 1
http://3g.163.com/news/17/0330/17/CGPSE27000018750.html 1
http://3g.163.com/news/17/0330/17/CGPTKEHI000187VE.html 1
http://3g.163.com/news/17/0330/17/CGPU6SO700018990.html 1
http://3g.163.com/news/17/0330/18/CGPV7ERR0001875O.html 1
http://3g.163.com/news/17/0330/18/CGPVL9OQ0001875O.html 1
http://3g.163.com/news/17/0330/18/CGPVO52P0001875M.html 1
http://3g.163.com/news/17/0330/18/CGPVTEIT00018AOR.html 1
http://3g.163.com/news/17/0330/18/CGQ0HNPF00018AOR.html 1

http://3g.163.com/news/17/0330/19/CGQ2UVML000187VE.html 1
```

```
http://3g.163.com/news/17/0330/19/CGQ3NSRI00018AOQ.html 1
http://3g.163.com/news/17/0330/19/CGQ3U5F00001899N.html 1
http://3g.163.com/news/17/0330/19/CGQ501NF00018AOR.html 1
http://3g.163.com/news/17/0330/19/CGQ5E9UR0001899O.html 1
http://3g.163.com/news/17/0330/20/CGQ6BIKG0001875P.html 1
http://3g.163.com/news/17/0330/20/CGQ74OVN0001899O.html 1
http://3g.163.com/news/17/0330/20/CGQ860RH0001875M.html 1
www.thepaper.cn 1
‘   1
’   1
“   39
”   39
、   52
。   181
。。。 1
《   2
》   2
【   1
】   1
〔   1
〕   1
一   25
一个  6
一些  2
一大早 1
一样  3
一直  1
一经  1
一起  3
一跃而起    1
一路  1
一边  1
一道  1
七   1
万   4
万一  1
丈夫  1
丈母娘 1
三   5
上   11
上午  5
上述  1
下   2
下午  2
下台  1
下同  4
下属  2
下来  1
下达  1
不   20
不予  1
不仅  2
不公  1
不动产 4

不同  1
```

```
不少 1
不幸 1
不得 3
不得了 1
不断 1
不满 2
不见 1
不过 2
与 8
与否 1
丑闻 1
专业 3
专家 1
专案组 1
专访 1
专门 2
且 1
东南亚 1
东经 1
东莞 2
东莞市 5
两 15
严厉 1
严格 2
严肃 1
严重 2
丧葬 1
...
...
...
...
采用 1
采访 2
释放 1
里 1
重 1
重大 4
重庆 2
重要 1
重视 1
金属 2
金秋 1
金章洙 2
金融 3
金额 1
鉴于 1
针对 4
钱 1
铁岭 1
铁岭市 4
铁路 2
银行业 2
销售 2
销售商 1

销毁 1
```

```
锄头 1
锅 1
长 1
长安街 1
问 3
问话 2
问询 3
问题 5
间 1
防控 1
陆慷 4
陆续 1
限制 3
限购 4
除 2
随即 1
随后 2
随着 1
随身 1
隐匿 1
隐隐 1
雅 2
雅万 3
雅万高铁 1
雇员 1
需 2
需要 6
震惊 1
震源 2
青瓦台 2
非常 2
非本市 2
靠 2
面 1
面向 1
面部 1
韩 6
韩国 22
韩媒 1
韩方 1
项 1
项目 8
顺差 1
须 5
预计 1
领域 1
领导 4
领导人 1
领导人员 1
风险 1
首 4
首尔 3
马 4
马方 2

马来西亚 8
```

```
驻华  2
骗  1
骗局  1
高  2
高于  1
高层  1
高强  2
高法  1
高铁  8
高院  4
鳌头人 1
鳌头镇 2
麻果  1
黄英  3
黑  1
黑人  1
！  6
！！！ 1
（  22
）  22
，  397
：  21
；  7
？  2
```