用户对物品的评分

	101	102	103	104	105	106
A	5	3	2.5	0	0	0
B	2	2.5	5	2	0	0
C	2	0	0	4	4.5	6
D	5	0	3	4.5	0	4
E	4	3	2	4	3.5	4

物品与物品的相似度

	101	102	103	104	105	106
101	5	3	4	4	2	3
102	3	3	3	2	1	1
103	4	3	4	3	1	2
104	4	2	3	4	2	3
105	2	1	1	2	2	2
106	3	1	2	3	2	3

用户对物品的评分矩阵 × 物品相似矩阵 = 推荐列表

构建物品相似度矩阵的时候可以通过计算两个物品的余弦相似度得出，于是需要构建每个物品在所有用户中的评分矩阵

	A	B	C	D	E
101	5	2	2	5	4
102	3	2.5	0	0	3
103	2.5	5	0	3	2
104	0	2	4	4.5	4
105	0	0	4.5	0	3.5
106	0	0	6	4	4

本例中，不采用余弦相似度的方式计算物品与物品相似度

在MapReduce作业中，输入数据的格式是：用户，物品，评分

```
A,101,5
A,102,3
A,103,2.5
B,101,2
B,102,2.5
B,103,5
B,104,2
C,101,2
C,104,4
C,105,4.5
C,106,6
D,101,5
D,103,3
D,104,4.5
D,106,4
E,101,4
E,102,3
E,103,2
E,104,4
E,105,3.5
E,106,4
```

第一步、构建用于评分矩阵，表示如下：

```
A    101:5,102:3,103:2.5
B    101:2,102:2.5,103:5,104:2
C    101:2,104:4,105:4.5,106:6
D    101:5,103:3,104:4.5,106:4
E    101:4,102:3,103:2,104:4,105:3.5,106:4
```

第二步、构建物品相似度矩阵。这里采用的方法是：如果两个物品同时出现在某个用户的评分矩阵中，则计数加1，例如，101和102同时出现在A B E中，因此101和102的相似度计为3，依次类推得出整个相似度矩阵，这个矩阵是一个对称矩阵。这一步的输入是第一步的输出。

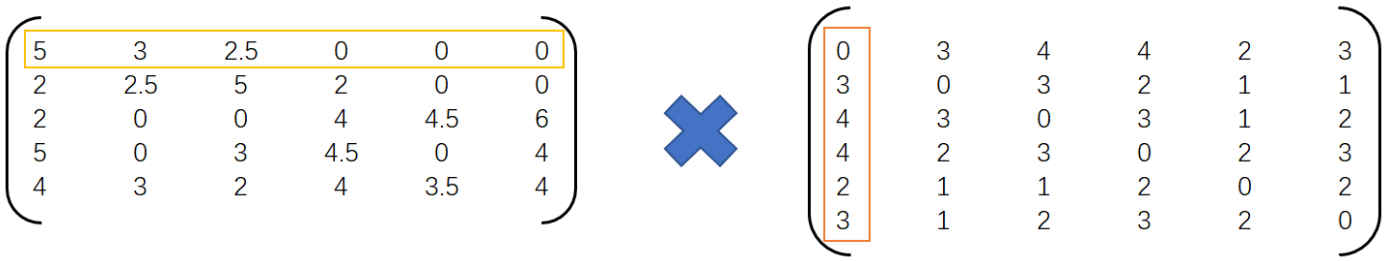
代码片段如下：

```java
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class Abc {

    public static class MatrixMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, Interru
            String[] terms = value.toString().split( regex: "[\\t,]");
            for (int i = 1; i < terms.length - 1; i++) {
                for (int j = i + 1; j < terms.length; j++) {
                    String a = terms[i].split( regex: ":")[0];
                    String b = terms[j].split( regex: ":")[0];
                    context.write(new Text( string: a + ":" + b), new IntWritable( value: 1));
                }
            }
        }
    }

    public static class MatrixReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();
        @Override
        protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOExcep
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) {
        String[] aa = {"A", "B", "C", "D"};
        for (int i = 0; i < aa.length - 1; i++) {
            for (int j = i + 1; j < aa.length; j++) {
                System.out.print(aa[i] + aa[j] + "  ");
            }
            System.out.println();
        }

    }

}
```

这样得出的结果类似于这样：

```
101:102     3
101:103     4
101:104     4
101:105     2
101:106     3
102:103     3
102:104     2
102:105     1
102:106     1
```

第三步、矩阵相乘。就是用第一步的输出矩阵乘以第二步的输出矩阵，这一步颇为复杂，需要将第二步的输出矩阵缓存起来



话不多说，上代码

workspace - [D:\workspace] - [hadoop-recommend] - ...\hadoop\hadoop-recommend\src\main\java\com\cjs\hadoop\goods\Step1.java - IntelliJ IDEA 2017.2.6

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

hadoop-recommend › src › main › java › com › cjs › hadoop › goods › Step1

Step1.java

```java
package com.cjs.hadoop.goods;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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 java.io.IOException;

/**
 * 计算评分矩阵
 */
public class Step1 {

    public static class Step1Mapper extends Mapper<LongWritable, Text, Text, Text> {
        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, Interrup
            String[] terms = String.valueOf(value.toString()).split( regex: ",");
            context.write(new Text(terms[0]), new Text( string: terms[1] + ":" + terms[2]));
        }
    }

    public static class Step1Reducer extends Reducer<Text, Text, Text, Text> {
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, In
            StringBuffer sb = new StringBuffer();
            for (Text val : values) {
                sb.append(",").append(val);
            }
            String res = sb.toString();
            if (res.length() > 1) {
                res = res.substring(1);
            }
            context.write(key, new Text(res));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration cfg = new Configuration();
        Job job = Job.getInstance(cfg,  jobName: "step1");
        job.setJarByClass(Step1.class);
        job.setMapperClass(Step1Mapper.class);
        job.setReducerClass(Step1Reducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path( pathString: "/input/m1.txt"));
        FileOutputFormat.setOutputPath(job, new Path( pathString: "/output/step1"));
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

4: Run 6: TODO 0: Messages Spring Java Enterprise Run Dashboard Terminal

Compilation completed successfully in 6s 109ms (today 10:37)

workspace - [D:\workspace] - [hadoop-recommend] - ...\hadoop\hadoop-recommend\src\main\java\com\cjs\hadoop\goods\Step2.java - IntelliJ IDEA 2017.2.6

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Step2.java

```java
package com.cjs.hadoop.goods;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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 java.io.IOException;
import java.util.Arrays;

/**
 * 计算相似矩阵
 */
public class Step2 {

    public static class Step2Mapper extends Mapper<LongWritable, Text, Text, IntWritable> {

        public static IntWritable one = new IntWritable( value: 1);

        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, Interrup
            String[] terms = value.toString().split( regex: "\\t")[1].split( regex: ",");
            Arrays.sort(terms); //  排序，避免重复
            for (int i = 0; i < terms.length; i++) {
                for (int j = 0; j < terms.length; j++) {
                    if (j == i) {
                        continue;
                    }
                    String a = terms[i].split( regex: ":")[0];
                    String b = terms[j].split( regex: ":")[0];
                    context.write(new Text( string: a + ":" + b), one);
                }
            }
        }
    }

    public static class Step2Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        @Override
        protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOExcept
            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 cfg = new Configuration();
        Job job = Job.getInstance(cfg,  jobName: "step2");
        job.setJarByClass(Step2.class);
        job.setMapperClass(Step2Mapper.class);
        job.setReducerClass(Step2Reducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path( pathString: "/output/step1"));
        FileSystem fs = FileSystem.get(cfg);
        fs.delete(new Path( pathString: "/output/step2"),  b: true);
        FileOutputFormat.setOutputPath(job, new Path( pathString: "/output/step2"));
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }

}
```

workspace - [D:\workspace] - [hadoop-recommend] - ...\hadoop\hadoop-recommend\src\main\java\com\cjs\hadoop\goods\Step3.java - IntelliJ IDEA 2017.2.6

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

hadoop-recommend > src > main > java > com > cjs > hadoop > goods > Step3

Step3.java

```java
package com.cjs.hadoop.goods;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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 java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.net.URI;
import java.util.*;

/**
 * 矩阵相乘
 */
public class Step3 {

    public static class Step3Mapper extends Mapper<LongWritable, Text, Text, Text> {

        private Map<String, Map<String, Integer>> similarMap = new HashMap<String, Map<String, Intege

        @Override
        protected void setup(Context context) throws IOException, InterruptedException {
            URI uri = context.getCacheFiles()[0];
            FileSystem fs = FileSystem.get(context.getConfiguration());
            BufferedReader br = new BufferedReader(new InputStreamReader(fs.open(new Path(uri))));
            String line = null;
            while (null != (line = br.readLine())) {
                String[] arr = line.split( regex: "[\\t:]");
                Map<String, Integer> map = similarMap.get(arr[0]);
                if (null == map) {
                    map = new HashMap<String, Integer>();
                    similarMap.put(arr[0], map);
                }
                map.put(arr[1], Integer.valueOf(arr[2]));
            }
            System.out.println("=====================================");
            System.out.println(similarMap);
            br.close();
        }

        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, Interru
            String[] ma = String.valueOf(value.toString()).split( regex: "\\t");
            String k = ma[0];
            String[] scores = ma[1].split( regex: ",");

            for (Map.Entry<String, Map<String, Integer>> entry : similarMap.entrySet()) {
                BigDecimal sum = new BigDecimal( val: 0);
                String col = entry.getKey();
                Map<String, Integer> map = entry.getValue();
                for (String score : scores) {
                    String kk = score.split( regex: ":")[0];
                    String vv = score.split( regex: ":")[1];

                    Integer s1 = map.get(kk);
                    if (null != s1) {
                        Double s2 = Double.valueOf(vv);
                        sum = sum.add(new BigDecimal(s1).multiply(new BigDecimal(s2)));
                    }
                }
                context.write(new Text(k), new Text( string: col + ":" + sum.toString()));
            }
        }
    }

    public static class Step3Reducer extends Reducer<Text, Text, Text, Text> {
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, I
            List<String> result = new ArrayList<String>();
            for (Text val : values) {
                result.add(val.toString());
            }
```

```
            Collections.sort(result);
            context.write(key, new Text(result.toString()));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration cfg = new Configuration();
        Job job = Job.getInstance(cfg, jobName: "step3");
        job.addCacheFile(new Path( pathString: "/output/step2/part-r-00000").toUri());
        job.setJarByClass(Step3.class);
        job.setMapperClass(Step3Mapper.class);
        job.setReducerClass(Step3Reducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path( pathString: "/output/step1"));
        FileSystem fs = FileSystem.get(cfg);
        fs.delete(new Path( pathString: "/output/step3"), b: true);
        FileOutputFormat.setOutputPath(job, new Path( pathString: "/output/step3"));
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

Step3

4: Run | 6: TODO | 0: Messages | Spring | Java Enterprise | Run Dashboard | Terminal

Compilation completed successfully in 6s 109ms (today 10:37)

```
[root@localhost hadoop-2.9.0]# hadoop fs -cat /output/step1/part-r-00000
A	101:5,102:3,103:2.5
B	101:2,102:2.5,103:5,104:2
C	106:6,105:4.5,104:4,101:2
D	106:4,103:3,101:5,104:4.5
E	104:4,105:3.5,106:4,101:4,102:3,103:2
[root@localhost hadoop-2.9.0]# hadoop fs -cat /output/step2/part-r-00000
101:102 3
101:103 4
101:104 4
101:105 2
101:106 3
102:101 3
102:103 3
102:104 2
102:105 1
102:106 1
103:101 4
103:102 3
103:104 3
103:105 1
103:106 2
104:101 4
104:102 2
104:103 3
104:105 2
104:106 3
105:101 2
105:102 1
105:103 1
105:104 2
105:106 2
106:101 3
106:102 1
106:103 2
106:104 3
106:105 2
[root@localhost hadoop-2.9.0]# hadoop fs -cat /output/step3/part-r-00000
A	[101:19.0, 102:22.5, 103:29, 104:33.5, 105:15.5, 106:23.0]
B	[101:35.5, 102:25, 103:21.5, 104:28.0, 105:15.5, 106:24.5]
C	[101:43.0, 102:24.5, 103:36.5, 104:35.0, 105:24, 106:27.0]
D	[101:42.0, 102:37.0, 103:41.5, 104:41, 105:30.0, 106:34.5]
E	[101:52.0, 102:33.5, 103:48.5, 104:47.0, 105:29, 106:38.0]
[root@localhost hadoop-2.9.0]#
```