

Федеральное агентство связи  
Федеральное государственное бюджетное образовательное учреждение высшего  
образования  
«Сибирский государственный университет телекоммуникаций и информатики»  
(СибГУТИ)

Отчёт по практической работе № 3

По дисциплине: «Распределенная обработка  
информации»

Тема: «MapReduce: анализ графов социальных сетей»

Выполнил:  
студент гр. МГ-101  
Лукошкин В. Ю.

Проверил:  
Профессор Кафедры ВС  
Курносов М.Г.

Новосибирск 2021

## Задание.

Задан граф пользователей Twitter с информацией о их подписчиках. Граф хранится в текстовом файле, каждая строка которого имеет следующий формат: `<user_id> <follower_id>`. Файл размещен на кластере: `/home/pub/hadoop/lab34/followers.db`. **Требуется:**

1. Вычислить распределение количества подписчиков (статистическое распределение выборки).
2. Определить TOP-50 пользователей по числу подписчиков.
3. Вычислить среднее количество подписчиков.

## Ход работы.

Для начала необходимо посчитать число подписчиков каждого пользователя из имеющегося файла. Получившийся набор будет являться входным набором для последующих заданий.

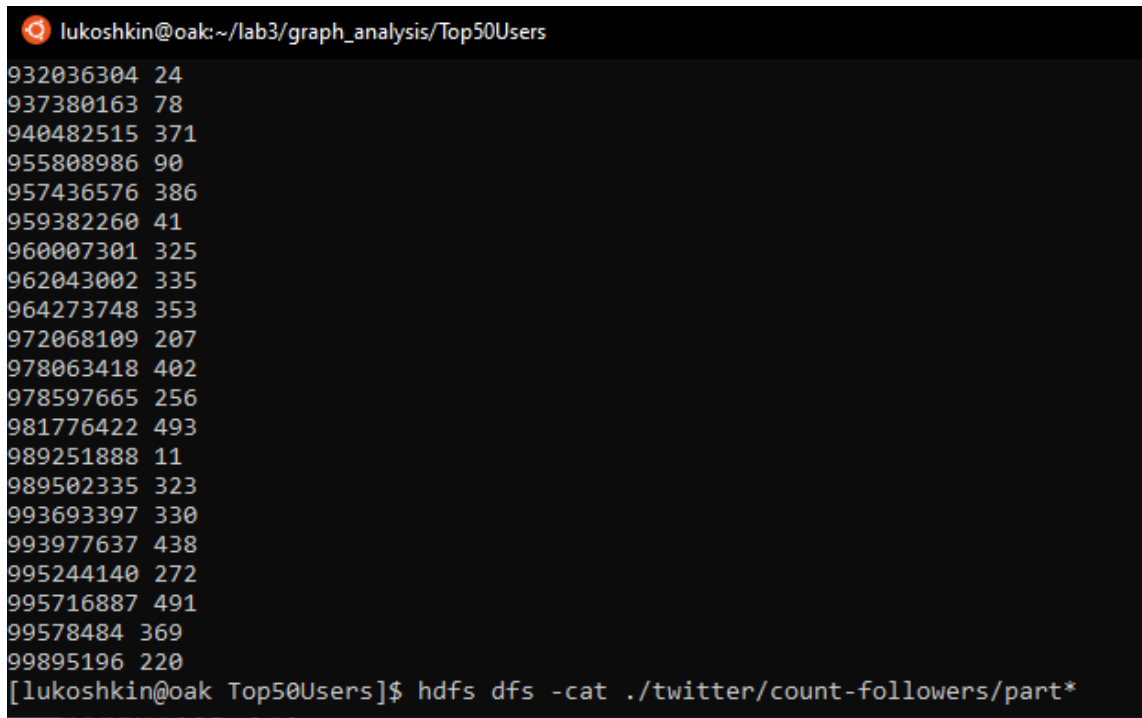
Копируем файл в HDFS:

```
[lukoshkin@oak CountFollowers]$ hdfs dfs -put /home/pub/hadoop/lab34/followers.db ./followers.db
put: `followers.db': File exists
[lukoshkin@oak CountFollowers]$
```

Проверим содержимое файла:

```
lukoshkin@oak:~/lab3/graph_analysis/Top50Users
1410691417 383948917
1410691417 186759612
1410691417 1988917809
1410691417 101996627
1410691417 1409724622
1410691417 1601837497
1410691417 1060283712
1410691417 1016982237
1410691417 530669988
1410691417 1791387173
1410691417 1178873856
1410691417 1243247309
1410691417 1750631862
1410691417 563186075
1410691417 1039227732
1410691417 1078252789
1410691417 1670184785
1410691417 427263908
1410691417 303519799
1410691417 1147495911
1410691417 1828597576
1410691417 801434465[lukoshkin@oak Top50Users]$ hdfs dfs -cat ./followers.db_
```

Компилируем CountFollowers.java, запускаем ./start-job.sh. В результате в HDFS будет создан каталог ./twitter/count-followers в котором содержится количество подписчиков для каждого пользователя.



```
lukoshkin@oak:~/lab3/graph_analysis/Top50Users
932036304 24
937380163 78
940482515 371
955808986 90
957436576 386
959382260 41
960007301 325
962043002 335
964273748 353
972068109 207
978063418 402
978597665 256
981776422 493
989251888 11
989502335 323
993693397 330
993977637 438
995244140 272
995716887 491
99578484 369
99895196 220
[lukoshkin@oak Top50Users]$ hdfs dfs -cat ./twitter/count-followers/part*
```

1. Вычислить распределение количества подписчиков (статистическое распределение выборки).

Компилируем DistributionCountFollowers.java, запускаем ./start-job.sh. Исходным набором будет файл из первого шага. Получившийся результат будет находится в HDFS в каталоге ./twitter/distribution-count-followers:

```
lukoshkin@oak:~/lab3/graph_analysis/CountFollowers
475 5
479 7
483 8
487 5
491 3
495 7
499 2
1051 1
1167 1
1591 1
1595 1
2195 1
3155 1
3227 1
3659 1
3687 1
4951 1
5155 1
5879 1
8435 1
9763 1
[lukoshkin@oak CountFollowers]$ hdfs dfs -cat ./twitter/distribution-count-followers/part*_
```

На выходе получили файл, где каждая строка имеет формат:

<follower\_count> <user\_count>.

2. Определить TOP-50 пользователей по числу подписчиков.

Компилируем Top50Users.java, запускаем ./start-job.sh. Исходным набором будет файл из первого шага. Получившийся результат будет находится в HDFS в каталоге ./twitter/top-50-users:

```
lukoshkin@oak:~/lab3/graph_analysis/Top50Users
1564013050 4772
916028859 4788
894879401 4829
1482851197 4951
241919610 4969
697527721 5155
729578616 5440
474623996 5857
2058667199 5879
202560399 5978
1407402292 6092
1131186229 6240
1984220012 6410
1564749221 7213
1559446157 7830
1060504695 8009
1917114697 8425
2107664819 8435
1820052348 8936
1767120751 9085
989240775 9763
[lukoshkin@oak Top50Users]$ hdfs dfs -cat ./twitter/top-50-users/part*_
```

На выходе получили файл, где каждая строка имеет формат:

<user\_id> <follower\_count>.

### 3. Вычислить среднее количество подписчиков.

Компилируем AvgCountFollowers.java, запускаем ./start-job.sh. Исходным набором будет файл из первого шага. Получившийся результат будет находится в HDFS в каталоге ./twitter/avg-count-followers/:

```
383.51813  
[lukoshkin@oak AvgCountFollowers]$ hdfs dfs -cat ./twitter/avg-count-followers/part*
```

В результате получилось, что среднее количество подписчиков составляет 383.

## Исходный код программы

### CountFollowers.java

```
package mapred.graphanalysis;

import java.io.BufferedReader;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.io.WritableUtils;
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.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.StringUtils;
```

```

import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class CountFollowers extends Configured implements Tool {

    /*
     * MAPPER
     */
    public static class CountMapper extends Mapper<Text, Text, Text,
IntWritable> {

        private long numRecords = 0;
        private Map<String, Integer> results = new HashMap<String,
Integer>();

        public void map(Text key, Text value, Context context) throws
IOException, InterruptedException {
            String user_id = key.toString();
            Integer count = results.get(user_id);
            if (count == null) {
                results.put(user_id, 1);
            } else {
                results.put(user_id, ++count);
            }

            if ((++numRecords % 1000) == 0) {
                context.setStatus("Finished processing " + numRecords +
" records");
                emitResults(context);
            }
        }

        private void emitResults(Context context) throws IOException,
InterruptedException {

```

```

        for (Entry<String, Integer> counts : results.entrySet()) {
            context.write(new Text(counts.getKey()), new
IntWritable(counts.getValue()));
        }
        results.clear();
    }

    @Override
    protected void cleanup(Context context) throws IOException,
InterruptedException {
        emitResults(context);
    }
}

/*
 * REDUCER
 */
public static class CountReducer 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);
    }
}

/*

```



```

    * APPLICATION
    */
    public int run(String[] args) throws Exception {
        Configuration conf = getConf();
        if (args.length != 2) {
            System.err.println("Usage: CountFollowers <input_path>
<output_path>");
            System.exit(2);
        }

        Job job = new Job(conf);
        job.setJarByClass(CountFollowers.class);
        job.setJobName("count followers");

        job.setMapperClass(CountMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //job.setCombinerClass(CountReducer.class);
        //job.setSortComparatorClass(BigramComparator.class);

        job.setReducerClass(CountReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setInputFormatClass(KeyValueTextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        boolean success = job.waitForCompletion(true);
        return success ? 0 : 1;
    }

```

```

    }

    public static void main(String[] args) throws Exception {
        int ret = ToolRunner.run(new CountFollowers(), args);
        System.exit(ret);
    }
}

```

### **DistributionCountFollowers.java**

```

package mapred.graphanalysis;

import java.io.BufferedReader;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;

```

```
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.io.WritableUtils;
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.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.StringUtils;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

```
public class DistributionCountFollowers extends Configured implements
Tool {
```

```
    /*
     * MAPPER
     */
```

```
    public static class CountMapper extends Mapper<Text, Text,
IntWritable, IntWritable> {
```

```
        private long numRecords = 0;
        private Map<Integer, Integer> results = new HashMap<Integer,
Integer>();
```

```
        public void map(Text key, Text value, Context context) throws
IOException, InterruptedException {
```

```
            Integer followers_count =
Integer.parseInt(value.toString());
            Integer count = results.get(followers_count);
            if (count == null) {
                results.put(followers_count, 1);
            } else {
```

```

        results.put(followers_count, ++count);
    }

    if ((++numRecords % 1000) == 0) {
        context.setStatus("Finished processing " + numRecords +
" records");

        emitResults(context);
    }
}

private void emitResults(Context context) throws IOException,
InterruptedException {
    for (Entry<Integer, Integer> counts : results.entrySet()) {
        context.write(new IntWritable(counts.getKey()), new
IntWritable(counts.getValue()));
    }
    results.clear();
}

@Override
protected void cleanup(Context context) throws IOException,
InterruptedException {
    emitResults(context);
}

}

/*
 * REDUCER
 */

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

```

```

        public void reduce(IntWritable 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);
        }
    }

    /*
    * APPLICATION
    */
    public int run(String[] args) throws Exception {
        Configuration conf = getConf();
        if (args.length != 2) {
            System.err.println("Usage: DistributionCountFollowers
<input_path> <output_path>");
            System.exit(2);
        }

        Job job = new Job(conf);
        job.setJarByClass(DistributionCountFollowers.class);
        job.setJobName("distribution count followers");

        job.setMapperClass(CountMapper.class);
        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(IntWritable.class);

        //job.setCombinerClass(CountReducer.class);
        //job.setSortComparatorClass(BigramComparator.class);

```

```

        job.setReducerClass(CountReducer.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);

        job.setInputFormatClass(KeyValueTextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        boolean success = job.waitForCompletion(true);
        return success ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int ret = ToolRunner.run(new DistributionCountFollowers(),
args);

        System.exit(ret);
    }
}

```

### **Top50Users.java**

```

package mapred.graphanalysis;

import java.io.BufferedReader;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.File;
import java.io.FileReader;

```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.StringTokenizer;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.io.WritableUtils;
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.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.StringUtils;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class Top50Users extends Configured implements Tool {

    /*
     * MAPPER
     */
}
```

```

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

        private TreeMap<Long, String> tmap;

        @Override
        public void setup(Context context) throws IOException,
InterruptedException {
            tmap = new TreeMap<Long, String>();
        }

        public void map(Text key, Text value, Context context) throws
IOException, InterruptedException {
            String user_id = key.toString();
            Long followers_count = Long.parseLong(value.toString());
            tmap.put(followers_count, user_id);
            if (tmap.size() > 50) {
                tmap.remove(tmap.firstKey());
            }
        }

        @Override
        public void cleanup(Context context) throws IOException,
InterruptedException {
            for (Map.Entry<Long, String> entry : tmap.entrySet()) {
                context.write(new Text(entry.getValue()), new
LongWritable(entry.getKey()));
            }
        }
    }

    /*
    * REDUCER
    */

```



```

        public static class CountReducer extends Reducer<Text,
LongWritable, Text, LongWritable> {
            private TreeMap<Long, String> tmap;

            @Override
            public void setup(Context context) throws IOException,
InterruptedException {
                tmap = new TreeMap<Long, String>();
            }

            public void reduce(Text key, Iterable<LongWritable> values,
Context context) throws IOException,
                InterruptedException {
                    String user_id = key.toString();

                    long count = 0;
                    for (LongWritable val : values) {
                        count = val.get();
                    }

                    tmap.put(count, user_id);
                    if (tmap.size() > 50) {
                        tmap.remove(tmap.firstKey());
                    }
                }

            @Override
            public void cleanup(Context context) throws IOException,
InterruptedException {
                for (Map.Entry<Long, String> entry : tmap.entrySet()) {
                    context.write(new Text(entry.getValue()), new
LongWritable(entry.getKey()));
                }
            }
        }

```

```

}

/*
 * APPLICATION
 */
public int run(String[] args) throws Exception {
    Configuration conf = getConf();
    if (args.length != 2) {
        System.err.println("Usage: Top50Users <input_path>
<output_path>");
        System.exit(2);
    }

    Job job = new Job(conf);
    job.setJarByClass(Top50Users.class);
    job.setJobName("top 50 users");

    job.setMapperClass(CountMapper.class);
    //job.setMapOutputKeyClass(Text.class);
    //job.setMapOutputValueClass(LongWritable.class);

    //job.setCombinerClass(CountReducer.class);
    //job.setSortComparatorClass(BigramComparator.class);

    job.setReducerClass(CountReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LongWritable.class);

    job.setInputFormatClass(KeyValueTextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

```

```
        boolean success = job.waitForCompletion(true);
        return success ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int ret = ToolRunner.run(new Top50Users(), args);
        System.exit(ret);
    }
}
```

## **AvgCountFollowers.java**

```
package mapred.graphanalysis;
```

```
import java.io.BufferedReader;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.StringTokenizer;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.io.WritableUtils;
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.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.StringUtils;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class AvgCountFollowers extends Configured implements Tool {

    /*
     * MAPPER
     */
    public static class CountMapper extends Mapper<Text, Text,
IntWritable, IntWritable> {

        private long numRecords = 0;
        private Map<Integer, Integer> results = new HashMap<Integer,
Integer>();

        public void map(Text key, Text value, Context context) throws
IOException, InterruptedException {
            //context.getCounter(Counters.INPUT_WORDS).increment(1);
            Integer followers_count =
Integer.parseInt(value.toString());
            Integer count = results.get(followers_count);
            if (count == null) {
                results.put(followers_count, 1);
            } else {
                results.put(followers_count, ++count);
            }

            if ((++numRecords % 1000) == 0) {
                context.setStatus("Finished processing " + numRecords +
" records");
            }
        }
    }
}

```

```

        emitResults(context);
    }
}

    private void emitResults(Context context) throws IOException,
InterruptedException {
        for (Entry<Integer, Integer> counts : results.entrySet()) {
            context.write(new IntWritable(counts.getKey()), new
IntWritable(counts.getValue()));
        }
        results.clear();
    }
    @Override
    protected void cleanup(Context context) throws IOException,
InterruptedException {
        emitResults(context);
    }
}

/*
 * REDUCER
 */
    public static class CountReducer extends Reducer<IntWritable,
IntWritable, Text, DoubleWritable> {

        static long val_followers;
        static long count;

        @Override
        public void setup(Context context) throws IOException,
InterruptedException {
            val_followers = 0;
            count = 0;
        }
    }

```

```

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

            val_followers += sum;
            ++count;
        }

```

```

        @Override
        public void cleanup(Context context) throws IOException,
InterruptedException {
            context.write(null, new DoubleWritable((double)
val_followers / count));
        }
    }

```

```

    /*
     * APPLICATION
     */
    public int run(String[] args) throws Exception {
        Configuration conf = getConf();
        if (args.length != 2) {
            System.err.println("Usage: AvgCountFollowers <input_path>
<output_path>");
            System.exit(2);
        }
    }

```

```

    Job job = new Job(conf);
    job.setJarByClass(AvgCountFollowers.class);
    job.setJobName("avg count followers");

    job.setMapperClass(CountMapper.class);
    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(IntWritable.class);

    //job.setCombinerClass(CountReducer.class);
    //job.setSortComparatorClass(BigramComparator.class);

    job.setReducerClass(CountReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(DoubleWritable.class);

    job.setInputFormatClass(KeyValueTextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    boolean success = job.waitForCompletion(true);
    return success ? 0 : 1;
}

public static void main(String[] args) throws Exception {
    int ret = ToolRunner.run(new AvgCountFollowers(), args);
    System.exit(ret);
}
}

```