Documentation

Z5136463

Zanlai Hu

1. Passing the values of args[0] and args[1]
   First, the code below should be put into the main function:

```
Configuration conf = new Configuration();
conf.set("numgram", args[0]);
conf.set("minicount", args[1]);
```

   Then, I can use the code below to get the value and use then in other classes or functions:

```
Configuration conf = context.getConfiguration();
String param = conf.get("numgram");
int numgram = Integer.parseInt(param);

Configuration conf = context.getConfiguration();
String param2 = conf.get("minicount");
int minicount = Integer.parseInt(param2);
```

2. Ngrams
   The Stringtokenizer function can separate sentence into words.
   So I create an array to store all the single words and use a for-loop to combine then based on the value of 'numgram':

```
int total = itr.countTokens();
String[] allword = new String[total];
int j = 0;
while (itr.hasMoreTokens()) {
                allword[j] = itr.nextToken();
                j++;
    }
//put all words from stringtoken to custom array.

for(int i = 0; i < total-numgram+1; i = i+1) {
                String givenword = allword[i];
                for(int k = 1; k < numgram; k++) {
                        givenword = givenword + " " + allword[i+k];
                }
```

```
word.set(givenword);
// question1: ngram. Using for-loop to make ngrams.
```

3. Minicount
   I override a new class called IntSumReducer2 to deal with the final merging step which used in setReducerClass:

```
job.setReducerClass(IntSumReducer2.class);
```

Then I use a if-statement to judge if the count is larger than minicount. If so, put it into result:

```
if(sum > minicount - 1) {
//Using if statement to judge if the count is bigger than minicount.
                    result.set(zuihou);
                    context.write(key, result);
                    }
```

4. Filename
   Combine the count and filename into a string and store it in the mapper step:

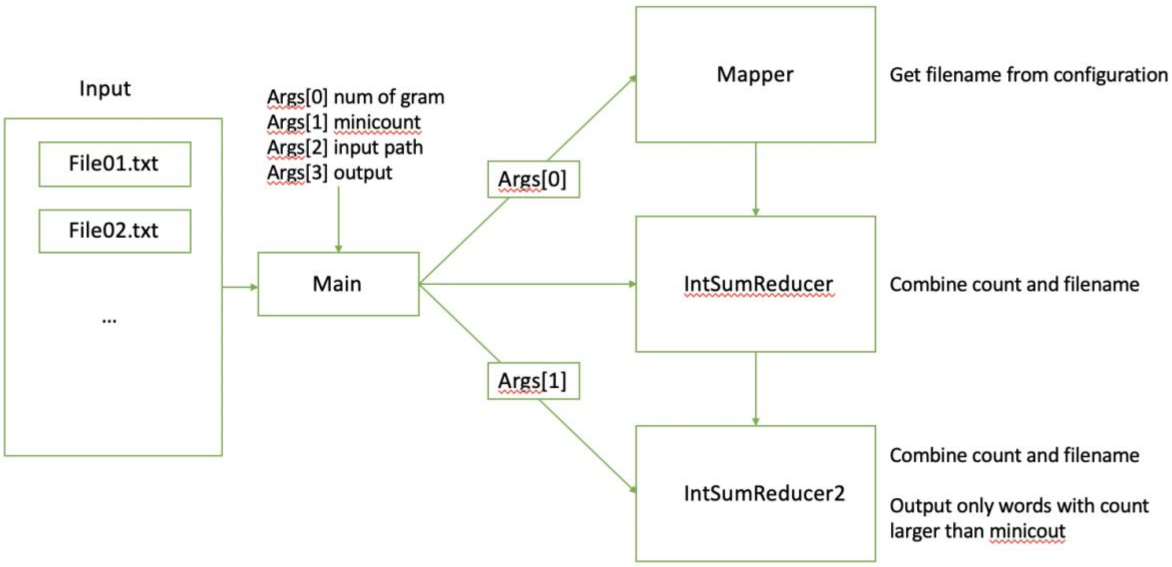
```
InputSplit inputSplit = context.getInputSplit();
String fileName = ((FileSplit) inputSplit).getPath().toString();
fileName = fileName.substring(fileName.length()-10, fileName.length());
one.set("1" + fileName);
```

Separate the count and filename in the reducer step:
For count, transfer the string into integer and sum it. Then transfer the integer to string. For filename, just combine them. Combine the sum and the final filename:

```
   String quzhi = new String();
   // Create a string to store the whole string.
   String wenjian = new String();
   // Create a string to store the filename.
   String zuihou = new String();
   // Create a string to put the final result.

   for (Text val : values) {

         quzhi = val.toString();
         int cishu = Integer.parseInt(quzhi.substring(0, 1));
         //Create an integer to calculate the sum.
```

```
sum += cishu;


wenjian = wenjian  + quzhi.substring(1, quzhi.length());
//combine all filenames for the same Text 'word'.
zuihou = Integer.toString(sum) + " " + wenjian;
//Put final result to string 'zuihou'.
  }
```

The flowchart of the code is showed as below:


Input
File01.txt
File02.txt
...
Args[0] num of gram
Args[1] minicount
Args[2] input path
Args[3] output
Main
Args[0]
Args[1]
Mapper
Get filename from configuration
IntSumReducer
Combine count and filename
IntSumReducer2
Combine count and filename
Output only words with count larger than minicout