Atitit 线程池使用范例 图像资料文档分类器 netpic image 网络图片与人像图片分类

[1.1. 微信图片分类 1](#_Toc30330)

[1.2. 手写日记 账单分类器 1](#_Toc6730)

[1.3. 线程池总共任务数量 ，待完成数量 ，已经完成数量 4](#_Toc20002)

[1.4. Json系列线程池对象时候去掉太长的queue属性+ 5](#_Toc25203)

[1.5. 记得关闭线程池最后 5](#_Toc11492)

## 微信图片分类

D:\0workspace\atiplat\_img\src\com\attilax\img\util\WechatPicFinder.java

## 手写日记 账单分类器

D:\0workspace\atiplat\_img\src\com\attilax\img\util\WechatPicFinderManaDiaryNCyarlogV2.java

package com.attilax.img.picClassifier;

import java.awt.Color;

import java.awt.image.BufferedImage;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.ThreadPoolExecutor;

import java.util.function.Function;

import com.alibaba.fastjson.serializer.JSONSerializer;

import com.alibaba.fastjson.serializer.PropertyPreFilter;

import com.alibaba.fastjson.serializer.SerializerFeature;

import com.attilax.img.HSV;

import com.attilax.img.HsvRangeV2;

import com.attilax.img.imgx;

import com.attilax.img.other.ColorUtil;

import com.attilax.io.dirx;

import com.attilax.io.filex;

import com.attilax.io.util.FileUtil\_tmp4img;

import com.attilax.json.AtiJson;

import com.attilax.util.FileUtil;

public class PicClassifierByNetpicNimgpic {

public static int num = 0;

public static void main(String[] args) {

// 28 16 88 , 40 12 85

// HSV hi = new HSV(200, 20, 95);

// sHSV low = new HSV(20, 0, 50);

float NetPicWhitepointCountLmt = 300; // percent

String strPath = "C:\\0000wechatpic";

strPath = "D:\\ati notvery impt\\r2017 cp pic bek v5 s229\\cp pic r2017 q4";

// + "//cp pic r2017 q2 v3 s223\\";

// String targetDir = "C:\\0000wechatpic\_tar\_NetInfoPic";

String targetDir = "D:\\ati notvery impt\\r2017 cp pic bek v5 s229\\cp pic r2017 q4 netpic incInfCyarOcr";

// + "\\cp pic r2017 q2 netpic incInfCyarOcr";

String strPath\_final = strPath;

ThreadPoolExecutor ExecutorService1\_theardpool = (ThreadPoolExecutor) Executors.newFixedThreadPool(4);

ExecutorService ExecutorService1=ExecutorService1\_theardpool;

dirx.traveV3(strPath, new Function<String, Object>() {

@Override

public Object apply(String cur\_f) {

num++;

System.out.println("num:" + num);

if (cur\_f.contains("IMG\_2017"))

return null; // jmp

ExecutorService1\_theardpool.submit(new Runnable() {

@Override

public void run() {

try {

createTask(NetPicWhitepointCountLmt, targetDir, strPath\_final, cur\_f);

} catch (Exception e) {

System.out.println(e);

}

}

}); // end sumbit

return null;

// end apply

}

private Object createTask(float wechatPicLmt, String targetDir, String strPath\_final, String cur\_f) {

System.out.println("tgg");

// only chat msg

// HSV hi = new HSV(95, 65, 95);

// HSV low = new HSV(85, 55, 85);

// all wechat pic

// HsvRangeV2 ran = new HsvRangeV2(low, hi);

// String f = "C:\\0000wechatpic\\a.png";

/\*

\* BufferedImage image = imgx.toImg(cur\_f); int heit =

\* image.getHeight(); int gray = 0; int hitCount = 0; for (int

\* curheit = 0; curheit < heit; curheit++) { for (int curwid =

\* 0; curwid < image.getWidth(); curwid++) { if (curwid == 609

\* && curheit == 699) System.out.println("dbg"); int clr\_int =

\* image.getRGB(curwid, curheit);

\*

\* Color cc2 = new Color(clr\_int); if ( ColorUtil.isWit(cc2)){

\* // if( isInSomeConditon()) hitCount++; }

\*

\* } } float pct = (float) hitCount / (heit \* image.getWidth())

\* \* 100; System.out.println("hitCount:" + hitCount + ",f:" +

\* cur\_f);

\*/

// System.out.println("pct:" + pct + ",f:" + cur\_f);

System.out.println("\_theardpool.getQueue().size(:" + ExecutorService1\_theardpool.getQueue().size());

// System.out.println("\_theardpool:

// "+AtiJson.toJson(ExecutorService1\_theardpool));

System.out.println("\_theardpool: "

+ com.alibaba.fastjson.JSON.toJSONString(ExecutorService1\_theardpool, new PropertyPreFilter() {

@Override

public boolean apply(JSONSerializer arg0, Object arg1, String prop) {

//// rt false //not show

if (prop.equals("queue"))

return false;

else

return true;

}

}, new SerializerFeature[] { SerializerFeature.PrettyFormat }));

// if (pct > wechatPicLmt) {

if (imgx.containsWhiteLine(cur\_f)) {

System.out.println("containsWhiteLine:" + "" + ",f:" + cur\_f);

// System.out.println("pct:" + pct + ",f:" + cur\_f);

filex.move(cur\_f, targetDir, strPath\_final);

// FileUtil\_tmp4img.copy(cur\_f, targetDir, strPath\_final);

}

return null;

}

});

ExecutorService1.shutdown();

System.out.println("--ExecutorService1.shutdown");

}

}

## 线程池总共任务数量 ，待完成数量 ，已经完成数量

**ThreadPoolExecutor** **ExecutorService1\_theardpool** = (**ThreadPoolExecutor**) **Executors**.*newFixedThreadPool*(4);

**System**.***out***.println( "\_theardpool.getQueue().size(:"+ExecutorService1\_theardpool.getQueue().size());

**System**.***out***.println("\_theardpool: "+**AtiJson**.*toJson*(ExecutorService1\_theardpool));

\_theardpool.getQueue().size(:1303

\_theardpool: {

"activeCount":4,

"completedTaskCount":264,

"corePoolSize":4,

"largestPoolSize":4,

"maximumPoolSize":4,

"poolSize":4,

"shutdown":false,

"taskCount":1571,

"terminated":false,

"terminating":false,

"threadFactory":{}

## Json系列线程池对象时候去掉太长的queue属性+

com.alibaba.fastjson.**JSON**.*toJSONString*(ExecutorService1\_theardpool, **new** PropertyPreFilter() {

***@Override***

**public** **boolean** **apply**(**JSONSerializer** arg0, **Object** arg1, **String** prop) {

//// rt false //not show

**if** (prop.equals("queue"))

**return** **false**;

**else**

**return** **true**;

}

}, **new** *SerializerFeature*[] { *SerializerFeature*.***PrettyFormat*** })

## 记得关闭线程池最后