Ruby 脚本实现数据爬取

作者：changming
原文链接：https://ld246.com/article/1471062902554
来源网站：链滴

工作上使用Redis，为了测试其性能，需要大量的测试数据，所以利用周末的时间用ruby写了个本，跑了一天，从搜搜问问和百度知道爬了大量的数据下来，分成问题和答案，总共有50万条数据，大小也就5M左右；周一去上班，写了一个ruby脚本连接Redis数据库，再循环分别导入问题和答案数据，这样试数据就有了，由于测试机器内存的限制，导入的测试数据总共100万条，占内存1G。
下面想说说ruby脚本的结构，很简单，还有需要改进的地方，但是脚本就是一次性的工具，如没通用性可能，有没有必要再改进另当其说。
建立http连接功能使用了内部包含的gem包open-uri，解析获取到的页面对象，并抓取特定的document元素，使用的gem是nkogiri，脚本分几个功能部分，各负其责，分别介绍如下：

递归抓取页面所有超级链接(spider_url.rb)


```
#!/usr/bin/ruby -w

require 'rubygems'
require 'nokogiri'
require 'open-uri'

load 'spider_document.rb'

class URL
attr_accessor :available_url, :pre_url, :visited_url, :error_url


def initialize
    @available_url = {}
    @visited_url = {}
    @error_url = {}
    @pre_url = "http://wenwen.soso.com"
end

# 抓取页面上的所有超链接，形如 'href...'
def crawl_url (target_url)
    puts '获取超级链接页面地址 -&gt; ' + target_url
    temp_available_url = {}

    begin
      open(target_url) do |uri|
        doc_content = uri.read
        doc_content.scan(/href=["|']](.+?)["|']]/) do |href_item|
          url = href_item[0]
                # TODO pattern is not exactely
        url.match(/.z./) do |m|

        # build hash {url=&gt;real url}
        if !@visited_url.has_key?(url)
          temp_available_url["#{url}"] = url
          @visited_url["#{url}"] = url
        end
```

```
        # puts '新增访问url：  ' + url
        end
        # url=@target_url.match(/(http:\/\/([^\/]+))\//)[1] &lt;&lt; url if url =~ /^\//
         end
       end
    rescue
      puts $!
      @error_url["#{target_url}"] = target_url
      puts 'error'
    end
    temp_available_url
end


def crawl_content (target_url)
    doc = Document.new
    doc.crawl_content(target_url)
end

end

url_spider = URL.new
puts 'url spider begining ...'

url_spider.available_url = url_spider.crawl_url("http://wenwen.soso.com")
while(!url_spider.available_url.empty?)
url_spider.available_url.each do |key,value|
url_spider.crawl_content(url_spider.pre_url + value)
url_spider.available_url = url_spider.available_url.merge url_spider.crawl_url(url_spider.pre_url
 value)
url_spider.available_url.delete(key)
puts 'current available_url size : ' + url_spider.available_url.size.to_s
end
end

puts 'Total available_url size : ' + url_spider.available_url.size.to_s
puts 'Total visited_url size : ' + url_spider.visited_url.size.to_s
puts 'Total error_url size : ' + url_spider.error_url.size.to_s<br /><br /></pre>

<ul>
<li><span>解析页面文档(</span><em><strong>spider_document.rb</strong></em><spa
>)</span></li>
</ul>
<pre class="brush: ruby">#!/usr/bin/ruby -w

require 'rubygems'
require 'nokogiri'
require 'open-uri'
```

```
class Document
@@quesion_count = 0
@@answer_count = 0


def get_question (page)
  questionArray = page.css('div.qa_title')
  questionArray.each do |question|
  #puts question.text
  File.open("question.txt",'a') { |f|
    f.puts question.text.strip.delete "快速回答".strip
  }
  end

 @@quesion_count = @@quesion_count + questionArray.size
 puts "current question count is : [" + @@quesion_count.to_s + "]"
end

def get_answer (page)
  answerArray = page.css('div.answer_con')
  answerArray.each do |answer|
    #puts answer.text
    File.open("answer.txt",'a') { |f|
      f.puts answer.text.strip.delete "快速回答".strip
    }
  end

  @@answer_count = @@answer_count + answerArray.size
  puts "current answer count is : [" + @@answer_count.to_s + "]"
end

def crawl_content (target_url)
   puts '抓取页面内容地址 -&gt; ' + target_url
   begin
     page = Nokogiri::HTML(open(target_url))
     get_question (page)
     get_answer (page)
   rescue Exception =&gt; e
        puts $!
   end
end

end    <br /><br /></pre>

<ul>
<li>批量导入Redis(<em><strong>spider_persistence.rb</strong></em>)</li>
</ul>
<pre class="brush: ruby">#!/usr/bin/ruby -w

require 'rubygems'
require 'nest'
require 'redis'
```

```
class Persistence


attr_accessor :redis, :question_count, :answer_count

def initialize
  @redis = Redis.new
  # @redis = Redis.new(:host=&gt;"192.168.1.67",:port=&gt;6379)
  @question_count = 0
  @answer_count = 0
end

# 批量生产账号
def batch_account
  account_namespace = Nest.new("account",@redis)

  File.open("account_email_local.txt") do |f|
    f.each_line do |line|
          pre_str = line.chomp.split('@')[0]
    account_namespace[line.chomp].hset("nickName",pre_str)
    account_namespace[line.chomp].hset("email",pre_str)
       account_namespace[line.chomp].hset("passWordHash","49") # 密码为1
    account_namespace[line.chomp].hset("answerCount","0")
    account_namespace[line.chomp].hset("selfDescription","非理性人类一枚")
    account_namespace[line.chomp].hset("followCount","0")
    account_namespace[line.chomp].hset("followerCount","0")
    account_namespace[line.chomp].hset("questionCount","0")

    puts line.chomp + " is builded."
    end
  end
end

# 批量生成问题集合
def batch_question
  account_namespace = Nest.new("account",@redis)
     question_namespace = Nest.new("question",@redis)
  pre_email = "rayootech" # 默认的账号 rayootech@163.com

  begin
       File.open("question.txt","r") do |file|
         file.each_line do |line|
    # 生成随机的20位问题id
    id = random_id(20)

    if (!line.strip.empty? &amp;&amp; line.strip.length&gt;3)
      puts "#{file.lineno} : #{line}"
      question_namespace[id].hset("id",id)
      question_namespace[id].hset("content",line)
      question_namespace[id].hset("author",pre_email+"@163.com")
      question_namespace[id].hset("createTime","2014-01-14")
      question_namespace[id].hset("followerCount","0")
      question_namespace[id].hset("browseCount","1")
```

```
        # 用户和提出的问题关系集合 account:[id]:question
      account_namespace["#{pre_email}@163.com"]["questions"].zadd(1401141645,id)
      @question_count = @question_count + 1
      File.open("question_id_local.txt", "a") { |f| f.puts id }
        end

      # 生成随机email地址前缀,并保存，后期生成account账号导入redis，一个email账户提500个问

      if (@question_count%500==0)
      pre_email = random_id(10)
      File.open("account_email_local.txt","a"){|file|file.puts "#{pre_email}@163.com"}
      end
     end
    end
    rescue Exception =&gt; e
     puts $!
    end
    end

# 批量生成回答集合
def batch_answer
    account_namespace = Nest.new("account",@redis)
    qa_relation_ns = Nest.new("question",@redis)
    answer_namespace = Nest.new("answer",@redis)
    question_id = "lzj4ggcgfpmj5uxnhtgx" # 【提问时间】 默认问题id

    begin
        File.open("answer.txt","r") do |file|
            file.each_line do |line|
      # 生成随机的20位回答id
      id = random_id(20)
      author = random_account_email
            if (!line.strip.empty?)
               puts "#{file.lineno} : #{line}"
      answer_namespace[id].hset("id",id)
      answer_namespace[id].hset("content",line)
      answer_namespace[id].hset("author",author)
      answer_namespace[id].hset("createTime","2014-01-15")
      answer_namespace[id].hset("approveCount","0")
      answer_namespace[id].hset("qId",question_id)

      # 问题和回答关系数据
      qa_relation_ns[question_id]["answers"].zadd(1401152040,id)
      # 问题的所有回答者关系数据
      qa_relation_ns[question_id]["respondents"].sadd(author)
      # 用户所有的回答数据
      account_namespace[author]["answers"].zadd(1401159088,id)

      @answer_count = @answer_count + 1
      File.open("answer_id_local.txt", "a") { |f| f.puts id }
            end

       # 每个问题下有平均100个回答
       if (@answer_count%100==0)
```

```
        question_id = random_question_id
          end

                end
    end
    rescue Exception =&gt; e
      puts $!
    end
end

# 批量生成问题浏览者集合
def batch_question_browser
end

# 随机返回一个问题id
def random_question_id
    question_id_arr = []
    index = 0
    File.open("question_id.txt") do |f|
      f.each_line do |line|
     question_id_arr[index]=line
     index = index + 1
      end
    end
    question_id_arr[rand(question_id_arr.size-1)].chomp
end

# 随机返回一个回答id
def random_answer_id
end

# 随机返回一个email
def random_account_email
    account_email_arr = []
    index = 0
    File.open("account_email.txt") do |f|
      f.each_line do |line|
     account_email_arr[index]=line
     index = index + 1
      end
    end
    account_email_arr[rand(account_email_arr.size-1)].chomp
end

# 生成随机数
def random_id(len)
       chars = ("a".."z").to_a + ("a".."z").to_a + ("0".."9").to_a
       random_id = ""
       1.upto(len) { |i| random_id &lt;&lt; chars[rand(chars.size-1)] }
       return random_id
  end

end
```

persistence = Persistence.new

1.times

puts "persistence question count : " + persistence.question_count.to_s

persistence.batch_account

1.times {|i| persistence.batch_answer }

puts "persistence answer count : " + persistence.answer_count.to_s</pre>