

HomeWork #3

Internet Bots, Spiders, and Emotional Agents

Fall 2001
Prof. Doug DeGroot <degroot@liacs.nl>

Homework 3: Email Harvesting

Goal: Harvest email addresses from files/web-pages; practice parsing in Rebol

Due: Thursday, October 11, 2001 (submit via email)

Description:

Shown below is a skeletal structure for a Rebol program that will read either a web page or a text file and then scan it for email addresses. The email addresses are collected into a list for later use in bulk-mailing applications. While most people will be familiar with this sort of application as it relates to the arena of spamming, it finds valuable use in Customer Service Relationship management as well, just to name one value-added application.

Here is the program skeleton:

```
REBOL [  
  Title: "Email Harvester 1.0"  
  Date: 23-Sept-2001  
  Author: "Doug DeGroot (so far!)"  
  File: %hw-email-harvesterDD.r  
  Purpose: {  
    Scan a web page or disk file for all possible email addresses.  
    Collect the email addresses for use in bulk mailing applications.  
  }  
  Comments: {  
    DD:  
    Make fault tolerant wrt read errors.  
    Extend the program so that it can also parse more email links.  
  }  
  Usage: {  
    Main functions are:  
    harvest-emaddds - scans a file or web page for email addresses  
    dump-the-emaddds - prints out the email addresses found  
  }  
  Category: [ spam-not! ]  
]  
  
verbose: on  
debugging: on
```

```

harvest-emadrs: func [theFile [url! file!] ] [
  em-list: copy [ ] ;initialize the global list of email addresses found
  txt: read theFile
  find-emadrs txt
  em-list: unique sort em-list
  vprint ["Found" length? em-list "email addresses."]
  if verbose [dump-the-emadrs]
  return em-list
]

find-emadrs: func [file] [
  parse file [any [thru "<mailto:" copy email-addr to ">"
    (handle-emaddr email-addr)] ]
  parse file [any [thru {href="mailto:} copy email-addr to {}"
    (handle-emaddr email-addr) ] ]
]

handle-emaddr: func [ema] [
; dprint ["Handling" ema]
; if <your test functions go here> [
;   dprint ["Adding email -- " ema]
;   append em-list ema
; ]
]

dump-the-emadrs: func [ ] [ foreach ema em-list [print ema] ]

```

Notice that the code shown here takes advantage of the utility functions I wrote about earlier --- namely, `dprint` and `vprint`. You should feel completely free to delete/replace these or any other parts of this program, including starting from scratch.

Problem:

As it is currently written, this program will recognize and collect only a limited number of email addresses. Furthermore, some of those that it collects will be “goofy” in some sense, as you will learn. This is due to the nature of the parsing rules in the *find-emadrs* function. Unfortunately, both web pages and other files contain email addresses embedded within the text in many formats.

Assignment:

Your assignment, should you choose to accept it, is to improve and extend this program so that it collects as many email addresses as possible from a given web page or text file – the more the merrier.

1. First, make the program fault tolerant with respect to reading the file or web page.

2. Then, run the program on a number of different web pages and see how many email addresses it finds. Examine the source code of those web pages to see if any were missed.
3. Do the same with a number of your other files, including email archives or Usenet newsgroup files.
4. Once you see what sorts of email addresses are missed by the program, either fix the existing parsing rules or add additional rules as needed in order to grab those extra email addresses.
5. Filter out any “junk” embedded in an email addresses that you think should be.
6. Notice that many email addresses are preceded by the person’s name, as in the following examples:

Bob Smith <bsmith@yahoo.com>
“Smith, Bob” <bsmith@yahool.com>

Capture the name, if there is one, along with the email address and store the names with the addresses. You can choose any data structure format you wish for the “database” of names and email addresses. Be sure to save the database to disk.

7. For extra credit, consider capturing and storing any other relevant information you find related to the email address. For example, if you are scanning an email discussion list, you could store the contents of the Subject: line along with the email address. This will clearly require some additional processing of the file.

Enjoy yourself!