Information Retrieval and Web Search

Crawling in practice

Instructor: Rada Mihalcea
Basic Crawl Architecture

WWW

DNS

Fetch

Parse

Doc FP’s

robots filters

URL set

Content seen?

URL filter

Dup URL elim

URL Frontier
Processing Steps in Crawling

• Pick a URL from the frontier
• Fetch the document at the URL
• Parse the URL
  – Extract links from it to other docs (URLs)
• Check if URL has content already seen
  – If not, add to indexes
• For each extracted URL
  – Ensure it passes certain URL filter tests
  – Check if it is already in the frontier (duplicate URL elimination)
Obtaining a webpage (lynx)

- Non graphical web browser
- It can dump the retrieved content in a clean text format
- It also provides a list of outgoing links
- Simple usage: `> lynx [options] [URL]`
- Example: `> lynx –dump www.google.com`
- Useful options:
  - *dump* – dump contents in text format
  - *timeout=N* – sets maximum timeout for waiting for a page
Lynx (continued)

• Benefits:
  – automatically solves DNS resolution
  – Provides already cleaned content
  – Completes relative addresses

• Drawbacks:
  – No access to metadata
  – No access to anchor text
  – Does not handle pages with frameset (it does not replace the frame content with the target pages)

• One line to get the links from a page:
  
  > lynx www.google.com -dump | grep -e "[0-9]*\.[ \n20http://.\n20]*"

• Available only in Linux/Unix environment
Obtaining a webpage (wget)

- Linux application designed to retrieve a webpage, or follow links up to a certain depth
- Simple Usage: `wget [option] [URL]`
- Useful options:
  - `-O file` (alternatively `--output-document=file`) – specify the target output document
  - `-r` - recursive downloading
  - `-l depth` - follow links only up to the depth of `depth` (default `depth=5`)
  - `-T seconds` - timeout
Wget (continued)

• Simple wget example:

  >wget www.google.com -O output

• Benefits:
  – Unmodified content (original format)
  – Recursive download
Perl LWP

- A collection of tools providing an API to the World Wide Web
- No need for external programs
- Fully portable and platform independent
LWP continued

# Create a user agent object
use LWP::UserAgent;
$ua = LWP::UserAgent->new;
$ua->agent('agentName');
    $ua->timeout(15);
# Create a request
    my $req = HTTP::Request->new(GET => 'http://www.google.com');
# Pass request to the user agent and get a response back
my $res = $ua->request($req);
# Check the outcome of the response
if ($res->is_success) { print $res->content; } 
    else { print $res->status_line, "\n"; }
Obtaining links

- Regular expressions
- HTML parsing via the HTML::Parse package
- HTML::Parse
  - Event based parser (scans through the document, and whenever finds an html tag, it generates an event and calls a predefined handler function)
  - We can overwrite handler functions
  - Flexible, customizable
  - We can extract both links and text content in one pass
HTML::Parse

- Important event handlers:
  - start – triggered when a start tag is found
  - end – triggered when an end tag is found
  - text – triggered when text information is found
  - comment – when comment is found

Example:
```html
<a href="http://www.google.com">Google</a>
```

Start event  text  end
import urllib2

url = “http://www.scrapy.org” #A scraping library
#Retrieve the data from the specified page
page = urllib2.urlopen(url)
#convert the data to a flat string
html = page.read()
#do all processing manually
from lxml import html
import requests

page = requests.get('http://lxml.de')  # lxml docs

# Create an HTML element object of the page
tree = html.fromstring(page.text)

# Iterate over the object’s links
for link in tree.iterlinks():
    print link #(element, attribute, link, pos)
Python BeautifulSoup

from bs4 import BeautifulSoup
import requests

url = "http://www.crummy.com/software/BeautifulSoup/bs4/doc/"
#Retrieve request result object from url
r = requests.get(url)
#Parse the text into structured HTML
soup = BeautifulSoup(r.text)

#Get all ‘href’-tagged elements from structured soup and print them
for link in soup.find_all('a'):
    print(link.get('href'))