MERC/IST 2013

Extracção e Análise de Dados na Web

Lab 05: Crawling the Web

The goal of this exercise is to implement a simple Web crawler.

1

Implement a crawler that takes as input a list of seed URL and collects Web pages starting from there. The collection should be done in a **breadth-first** manner. Each collected page should be stored in a separate HTML file.

Notes:

- We can use any naming convention you wish for the files (e.g. use a unique number for each file);
- To get a page from the Web you can use the `urllib2` module. For example:

```
from urllib2 import urlopen
site = urlopen("http://www.ist.utl.pt")
content = site.read()
print content
site.close()
```

- You can collect anchor links from the HTML page using regular expressions. For example:

```
import re
linksre = '<a\s.*?href=[\'"](.*?)[\'"].*?</a>'
links = re.findall(linksre, content, re.I)
```

- You can use the `urlparse` module to transform relative links into absolute links. For example:

```
import urlparse
url = urlparse.urljoin("http://www.ist.utl.pt/", "eventos/")
print url
```

- After transforming the links to absolute links, consider only those that start with "http";
- Do not worry about transforming the URL into their canonical form;
- Make sure you do not collect the same link twice;
- You will want to **limit the depth** of the collection;

- Make sure you wait at least one second before each server request. For example, you can use the `time` module:

  ```
  import time
  time.sleep(1)
  ```

- You can use the `robotparser` module to interpret the *robots.txt* file. For example:

  ```
  import robotparser
  rp = robotparser.RobotFileParser("http://www.ist.utl.pt/robots.txt")
  rp.read()
  print rp.can_fetch("*", "http://www.ist.utl.pt/pt/candidatos/")
  print rp.can_fetch("*", "http://www.ist.utl.pt/newscache/")
  ```

  Note that the *robots.txt* file usually only exists at the root of the server being accessed. The `RobotFileParser` class will not check if the file exists.

- **Remember:** some servers may block you, if you are not nice!

2

Modify you crawler, to create a vertical crawler. It should take as input a list of keywords, representing a topic (e.g. "peer to peer networks") and collect only pages within that topic.

To decide if a page is related to the given topic, you can simply count how many of the topic words it contains and set a decision threshold (e.g. if it contains at least 2/3 of the topic words, it should be collected).

3

Note: this item is optional and will not be evaluated.

Index the collected pages using Whoosh. Make sure you store the URL of each page. You may need to modify your crawler, to also store the URL.

Create a script that allows a user to perform searches. The result of a search should be a list of URL, sorted according to the page relevance. Together with each URL, there should be a text snippet for the page.

See the Whoosh documentation on how to present text snippets, at `http://pythonhosted.org/Whoosh/highlight.html`.