IST 2018/2019

Processamento e Recuperação de Informação

Lab 08: 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

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://whoosh.readthedocs.io/en/latest/highlight.html`.

4 Pen and Paper Exercises

(a) Compute the Jaccard similarity of each pair of the following sets:

- $\{1, 2, 3, 4, 5\}$

- $\{1, 6, 7\}$
- $\{2, 4, 6, 8\}$

(b) Suppose that you want to use the min-hash scheme for representing sets of items, in which there are ten different items that can be used within the sets (i.e., the universal item set is $\{1, 2, \ldots, 10\}$). Suppose also that the min-hash signatures for the sets are constructed using the following list of permutations for the universal set:

- $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$
- $(10, 8, 6, 4, 2, 9, 7, 5, 3, 1)$
- $(4, 7, 2, 9, 1, 5, 3, 10, 6, 8)$

Construct minhash signatures for the following sets:

- $\{3, 6, 9\}$
- $\{2, 4, 6, 8\}$
- $\{2, 3, 4\}$

(c) Suppose that instead of using particular permutations to construct the min-hash signatures for the three sets of the previous problem, we use an efficient single-pass implementation leveraging hash functions to construct the signatures. The three hash functions we use are as follows:

- $f(x) = x \mathrm{mod} 10$
- $g(x) = (2x + 1) \mathrm{mod} 10$
- $h(x) = (3x + 2) \mathrm{mod} 10$

Compute the signatures for the three sets, and compare the resulting estimate of the Jaccard similarity of each pair with the true Jaccard similarity.

(d) The function $p = 1 - (1 - s^r)^b$ gives the probability p that two min-hash signatues that come from sets with Jaccard similarity s will hash to the same bucket at least once, if we use an LSH scheme with b bands of r rows each. For a given similarity threshold s, we want to choose b and r so that $p = 1/2$ at s. Suppose signatures have length 24, which means we can pick any integers b and r whose product is 24 (i.e., the choices for r are 1, 2, 3, 4, 6, 8, 12, or 24, and b must then be $24/r$). If $s = 1/2$, determine the value of p for each choice of b and r, and state which value would you choose for r to maximize result quality.