Web Mining:
Assignment IV - A Simple Topical Web Crawler

Reykjavik University – School of Computer Science

Fall 20131

1 About the project

The goal of this project is to reinforce your understanding of web crawling by developing a simple web crawler in Java. Your web crawler should be a *sequential*, *topical* crawler, i.e. a crawler that does not use concurrency and which is activated in response to particular information needs. Your crawler starts from a single *seed* page and crawls a maximum number of pages it is allowed to visit.

The crawler is started in the following manner:

```
java WebCrawler <URL> <TOPIC> <QUERY WORDS> <N>
```

where:

1. *URL* is the seed to start the crawl.
2. *TOPIC* is the topic we are interested in (used to quide the crawler to relevant links).
3. *QUERY WORDS* is the phrase we are interested in.
4. *N* (optional) is the maximum number of pages to crawl.

The crawler uses a priority queue for unvisited links. The links are scored with 1.0 if the topic is part of the anchor text for a link (lower case comparison), else 0.0. It then visits links based on this priority score. The crawler should report on the number of pages that contain the phrase query (lower case comparison).

You are given a skeleton of a web crawler, consisting of six Java classes, and your task is to fill in the gaps, thus developing a functioning crawler. Gaps in the code, for each source file, are marked with special comments.

2 The source files

1. `WebCrawler.java`: This is the main class and contains the overall logic for the crawling. You have to fill in some gaps in this code.
2. `Frontier.java`: This class encapsulats the frontier. You have to fill in some gaps in this code.
3. `URLScore.java`: A data object containing a URL and its score. Yo do not have to change this class.

4. `HTMLParser.java`: This class provides methods to connect to an URL, to retrieve the links and to retrieve the body of an HTML document. It uses the `jsoup` JAVA HTML Parser, which you need to study – see `http://jsoup.org/`. You have to fill in some gaps in this code.

5. `URLCanonicalizer.java`: This class canonicalizes a given URL. You have to fill in some gaps in this code.

6. `RobotTxtParser.java`: This class supports the reading and checking of entries from a robot.txt file. You do not have to change this class.

3 Input and output

Your web crawler accepts four command line parameters: i) a seed page, a topic, a phrase query, and a number indicating the maximum number of pages to crawl.

The output is information printed to standard out, i.e. information about the given parameters, the pages containing the phrase query, and finally total number of pages containing the phrase query and the total number of distinctive links found.

Here is is an example output:

```
-------------------------------------------------------
Starting crawl, seed: http://mbl.is/
Topic: golf
Query string: birgir leifur
Maximum number of pages to visit: 100
-------------------------------------------------------
Query found in page: http://mbl.is/sport/golf/
Query found in page: http://mbl.is/sport/golf/2013/11/06/hoggi_fra_ad_komast_afram/
Query found in page: http://mbl.is/sport/golf/2013/11/06/vissa_hja_birgi_leifi/
Query found in page: http://mbl.is/sport/golf/2013/11/05/frestad_hja_birgi_leifi_sem_a_eina_holu_eftir/
Query found in page: http://mbl.is/sport/golf/2013/11/04/birgir_leifur_lek_a_einu_undir_pari/
Query found in page: http://mbl.is/sport/golf/2013/11/03/birgir_leifur_tharf_ad_leika_betur/
Query found in page: http://mbl.is/sport/golf/2013/10/25/birgir_leifur_komst_afram/
Query found in page: http://mbl.is/sport/golf/2013/10/25/birgir_leifur_a_tveimur_yfir_eftir_thrja_hringi/
Query found in page: http://mbl.is/sport/golf/2013/10/24/birgir_leifur_a_pari_eftir_tvo_hringi/
Query found in page: http://mbl.is/sport/golf/2013/10/22/birgir_leifur_lek_a_hoggi_undir_pari/
-------------------------------------------------------
Search complete, 100 pages crawled
Search query birgir leifur found in 10 pages
Total distinctive urls found: 936
-------------------------------------------------------
```

4 What to hand in

You need to hand in the following items:

1. All source code.

2. A *jar* file, containing an archive of the java classes files.

3. A script that runs you crawler using the .jar file. Put comments in the script directing the user what parameters should be given.

4. An example output, given a specific run of the script.

5 Additions

Feel free to add more functionality to your crawler, such as:

1. Allowing the user to specify a certain depth to be crawled.
2. Saving relevant pages in a repository.
3. Implemeting more sophisticated scoring mecanism for links.
4. Implementing concurrency.

Any addition should be properly commented.