Discussion 3: crawler4j

Jan 22\textsuperscript{nd}, 2014

Content adapted from \url{http://code.google.com/p/crawler4j/}
Recall: a robust crawl architecture

* Adapted from Jan 21 lecture
Crawler4j makes it easy

Configure your crawler
  local storage folder
  number of crawlers
  max depth/pages
  Politeness
  user agent string
  Proxy
  Resumable
  add seed

Start crawling
  while (fetch next url from frontier)
    If(this page should be visited)
      Extract data from this page
      Process data
      Extract outgoing links from this page
      Add links to frontier
Crawler4j makes it easy

Configure your crawler
  local storage folder
  number of crawlers
  max depth/pages
  Politeness
  user agent string
  Proxy
  Resumable
  add seed

Start crawling
  while (fetch next url from frontier)
    If(this page should be visited)
      Extract data from this page
      Process data
      Extract outgoing links from this page
      Add links to frontier
implement a crawler

• Extends from WebCrawler class and override two methods
  
  – boolean shouldVisit(WebURL url);
    
    • this function determines if a given url should be crawled (based on your own logic)
  
  – void visit(Page page);
    
    • This function is where your processing happen
      – Build index, record page statistics

• Outgoing links are added to frontier by crawler4j
create a CrawlController

```java
public class Controller {
    public static void main(String[] args) throws Exception {
        String crawlStorageFolder = "\data\crawl\root";
        int numberOfCrawlers = 7;

        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(crawlStorageFolder);

        /*
         * Instantiate the controller for this crawl.
         */
        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);

        /*
         * For each crawl, you need to add some seed urls. These are the first
         * URLs that are fetched and then the crawler starts following links
         * which are found in these pages
         */
        controller.addSeed("http://www.ics.uci.edu/~welling/");
        controller.addSeed("http://www.ics.uci.edu/~lopes/");
        controller.addSeed("http://www.ics.uci.edu/");

        /*
         * Start the crawl. This is a blocking operation, meaning that your code
         * will reach the line after this only when crawling is finished.
         */
        controller.start(MyCrawler.class, numberOfCrawlers);
    }
}
```

A local folder for intermediate crawl data

INF 141 / CS 121  Tao Wang
create a CrawlController

```java
public class Controller {
    public static void main(String[] args) throws Exception {
        String crawlStorageFolder = "/data/crawl/root";
        int numberOfCrawlers = 7;

        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(crawlStorageFolder);

        /*
         * Instantiate the controller for this crawl.
         */
        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);

        /*
         * For each crawl, you need to add some seed urls. These are the first
         * URLs that are fetched and then the crawler starts following links
         * which are found in these pages
         */
        controller.addSeed("http://www.ics.uci.edu/~welling/");
        controller.addSeed("http://www.ics.uci.edu/~lopes/");
        controller.addSeed("http://www.ics.uci.edu/");

        /*
         * Start the crawl. This is a blocking operation, meaning that your code
         * will reach the line after this only when crawling is finished.
         */
        controller.start(MyCrawler.class, numberOfCrawlers);
    }
}
```
create a CrawlController

```java
public class Controller {
    public static void main(String[] args) throws Exception {
        String crawlStorageFolder = "/data/crawl/root";
        int numberOfCrawlers = 7;

        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(crawlStorageFolder);

        /*
         * Instantiate the controller for this crawl.
         */
        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);

        /*
         * For each crawl, you need to add some seed urls. These are the first
         * URLs that are fetched and then the crawler starts following links
         * which are found in these pages
         */
        controller.addSeed("http://www.ics.uci.edu/~welling/");
        controller.addSeed("http://www.ics.uci.edu/~lopes/");
        controller.addSeed("http://www.ics.uci.edu/");

        /*
         * Start the crawl. This is a blocking operation, meaning that your code
         * will reach the line after this only when crawling is finished.
         */
        controller.start(MyCrawler.class, numberOfCrawlers);
    }
}
```

Nothing needs to be changed here
create a CrawlController

```java
public class Controller {
    public static void main(String[] args) throws Exception {
        String crawlStorageFolder = "data/crawl/root";
        int numberOfCrawlers = 7;

        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(crawlStorageFolder);

        /*
         * Instantiate the controller for this crawl.
         */
        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);

        /*
         * For each crawl, you need to add some seed urls. These are the first
         * URLs that are fetched and then the crawler starts following links
         * which are found in these pages
         */
        controller.addSeed("http://www.ics.uci.edu/~wellings/");
        controller.addSeed("http://www.ics.uci.edu/~lopes/");
        controller.addSeed("http://www.ics.uci.edu/");

        /*
         * Start the crawl. This is a blocking operation, meaning that your code
         * will reach the line after this only when crawling is finished.
         */
        controller.start(MyCrawler.class, numberOfCrawlers);
    }
}
```

Here are your url seeds
other configurations

• Maximum crawl depth: default is -1 for unlimited depth.
• A -> B -> C -> D: A has depth 0. Max depth = 2 means D won’t be crawled
other configurations

- Maximum number of pages to crawl: default is no limit
other configurations

- Politeness
other configurations

• User agent string: used for representing your crawler to web services. Default is
  “crawler4j (http://code.google.com/p/crawler4j/)”.

• To change:
  crawlConfig.setUserAgentString(userAgentString);
other configurations

• Proxy

* if you need to use proxy
  
  config.setProxyHost("proxyserver.example.com");
  config.setProxyPort(8080);

* If your proxy also needs authentication:
  
  config.setProxyUsername(username);
  config.getProxyPassword(password);
other configurations

- Resumable crawling
  - If your crawler will run for a long time
  - Possible unexpected termination
  - Resume from a previously stopped/crashed crawl

    crawlConfig.setResumableCrawling(true);
Other issues

• robots.txt
  – robotstxtServer.allows(webURL): check if a url is allowed to be crawled
  – Details of how crawler4j finds robots.txt
    RobotstxtServer.fetchDirectives(URL url);

• Duplicated urls
  – WebCrawler.processPage(WebURL curURL);
  – Relies on a docid. Details are in class DocIDServer.
learn more about crawler4j

  - All content in this presentation is adapted from this site
  - Limited documentation on the site
  - Source code available
    - git repository: [https://crawler4j.googlecode.com/git/](https://crawler4j.googlecode.com/git/).
  - Download samples:
    [https://crawler4j.googlecode.com/archive/e14a296409390eaba34108481b2ce779e0d99bbf.zip](https://crawler4j.googlecode.com/archive/e14a296409390eaba34108481b2ce779e0d99bbf.zip)
    - Crawler4j source code is available in the sample package
Discussion 3: crawler4j

Jan 22\textsuperscript{nd}, 2014

Content adapted from http://code.google.com/p/crawler4j/