**CAP6777 Web Mining**

Homework 1: Web crawler [15 pts, Due: 05/31/2016]

**Part 1:** Question and Answers [0.75 pt/each].

1. *What is HTTP? What is HTML? What are their relationships, and how are they related to web mining?*

* HTTP is a protocol, an agreement or communication standard between PC clients and webserver. HTML is a structure or a frame that is consisted of a set of text commands that can display a content of web pages. Also this set of text commands can be accessible and understandable by HTTP protocol so that any clients use HTTP communication can see and read web page. Since web mining is also another way to access to web page through the HTTP protocol thus HTTP and HTML are the two important objects that web mining has to understand them and follow their setting.

1. *What is a web crawler? What is the taxonomy of the web crawler? Please list major components of a web crawler*.

* Web crawler is a program that is built (written) to access or browse web pages automatically in order to collect a desired data. The taxonomy of the web crawler is a scheme of classification to designate to job (purpose or objective) that a web crawler is built for.

Major components of a web crawler: (<http://nlp.stanford.edu/IR-book/pdf/20crawl.pdf>)

The URL frontier

A DNS resolution module that determines the web server

A fetch module that uses the http protocol to retrieve the web page

A parsing module to extract the text

A duplicate elimination module to fill those URLs that have already been fetched.

1. *To develop a large scale universal crawler, the two major issues include “performance” and “policy”, please identify at least three components on how to improve the performance and how to design good policy, respectively*.

* Minimize overhead of DNS lookups.
* Optimize utilization of network bandwidth and disk throughput.
* Use asynchronous sockets to void ‘traffic jam’ to those websites that are mined.

1. *What is a spider trap (or crawler trap) in web crawling? Please suggest at least three approaches (heuristics) for a web crawler to identify a spider trap*.

* Spider trap is a set of web pages that creates an infinite number of URLs to make unintentionally or intentionally a web crawler crash or perform poorly.
* Three approaches for a web crawler to identify a spider trap:
* Check to see if a website has a deep directory structures.
* Check dynamic pages that produce an unbounded number of documents for a web crawler to follow.
* Check to see if documents filed with a large number of characters.

1. *What is Robots Exclusion Standard? Please use a real-world robots.txt example (e.g.,* [*http://www.google.com/robots.txt*](http://www.google.com/robots.txt)*) to explain the key fields (allow, disallow, sitemap etc.) and the setting of the robots exclusion protocol*.

* The robots exclusion standard is a standard or a policy that websites use to communicate with web crawlers to protect themselves (websites) from being over used (abused) or harmed that affects their activities.
* The file named ‘robots.txt’ in a server specifies which parts of the sites are allowed or disallowed to scan. Web crawler has to follow those policies.

The example of robots.txt

User-agent: \* => allow all robots

Disallow: /search => all robots not to enter ***search*** directory

Allow: /search/about => all robots can enter ***search/about*** directory

Disallow: /groups => all robots not to enter ***groups*** directory

Disallow: /index.html? => all robots not to enter ***index*** directory

Allow: /?hl= => all robots can enter ***/?hl=*** directory

User-agent: Twitterbot =>allow Twitter and FB to enter ***imgres*** directory

Allow: /imgres

User-agent: facebookexternalhit

Allow: /imgres

=>inform search engines about these sites (URLs) are available for crawling

Sitemap: http://www.gstatic.com/s2/sitemaps/profiles-sitemap.xml

Sitemap: https://www.google.com/sitemap.xml

1. *What is Cloaking and Spamdexing? Please explain how they are used to disguise the webservers and affect a search engine’s ranking*.

* Cloaking is an act of showing different content or URLs to human users and search engines. Cloaking is done by delivery content based on geography (IP addresses) or based on cookies history of users.
* Spamdexing is the practice of creating websites that will be illegitimately indexed with a high position in the search engines or to create a web page that will find favorable rankings in the search engines. Spamdexing is done by creating superfluous backlinks(linking to higher ranked pages) and creating spam content – articles or comments that contain nonsense to a reader, but search engines will see relevant keywords. (<http://www.webspam.org/seo-spam-what-is-spamdexing/>)

1. *What is “frontier” in the web crawler? Please suggest two approaches to manage the frontier, and explain the strength and possible weakness of these approaches*.

* Frontier (the URL frontier) is a component which is used to contain URLS yet to be fetched in the current crawl.
* Two approaches to manage it:
  + Since the frontier can grow very fast, high-quality pages should be prioritized for frequent crawling. So skip unwanted files by using extensions, however, this way might not be reliable because the file type can be various and changing quickly.
  + Issuing ‘HEAD’ HTTP commands to get Content-Type headers, however overhead of extra internet requests

**Part 2:** Simple web crawling practice [6 pts]

1. *Please download the java file, and use Eclipse or Netbean to build a web crawling project. Please provide a seed url and collect at least 50 web pages from the web crawler [1 pt]*
   1. *Please submit the original seed URL, and also include all web pages (in one file).*

I ran the java file and seed the URL: [www.voanews.com](http://www.voanews.com)

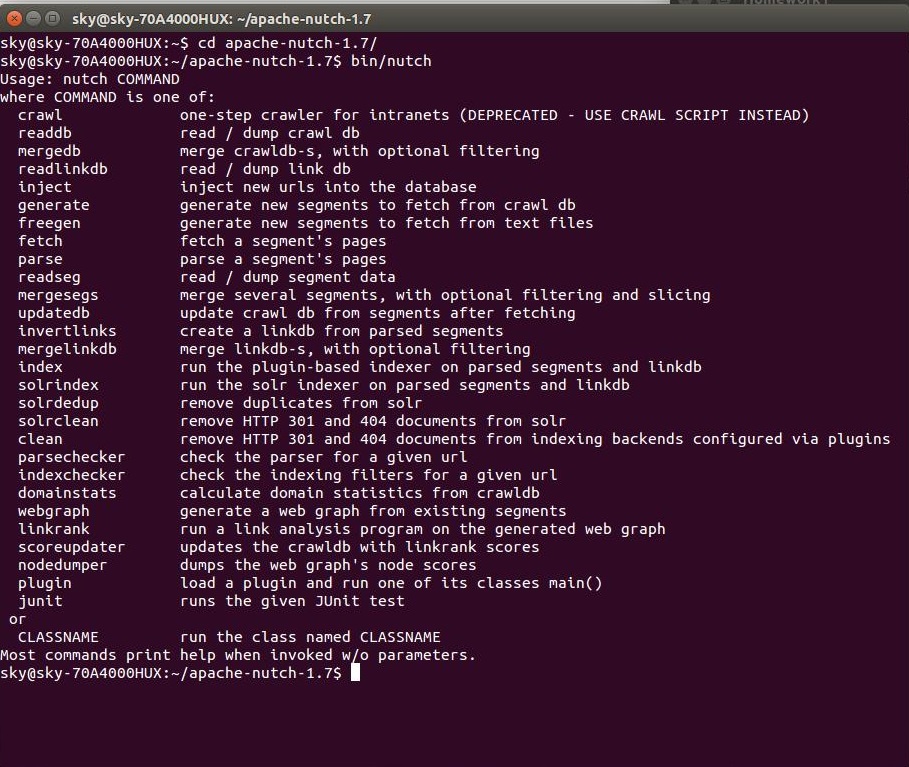
See the output Part2aVOANEWS.txt for the result of 50 websites

1. In the
   1. Please draw the flow chart (or the pseudo code) of your preferential web crawler design [1 pt], and explain how does your approach/design make web crawling focusing on special topics [1 pt].
   2. Please turn in the revised source code of your program [1 pt]
   3. Given the same set of seed URLs, please use original Webcrawler.java to collect 200 web pages, and also use your new preferential web crawler to collect 200 web pages. For 200 web pages collected from different web crawlers, please check the percentage of the web pages containing the specific keywords, and report the values in your report [1 pt].
   4. Please suggest one additional approach (show your design as a flow chart or a diagram) which may help improve the accuracy of preferential web crawler, so the collected documents are closely related to the topics [1 pt] (No need to implement this design).

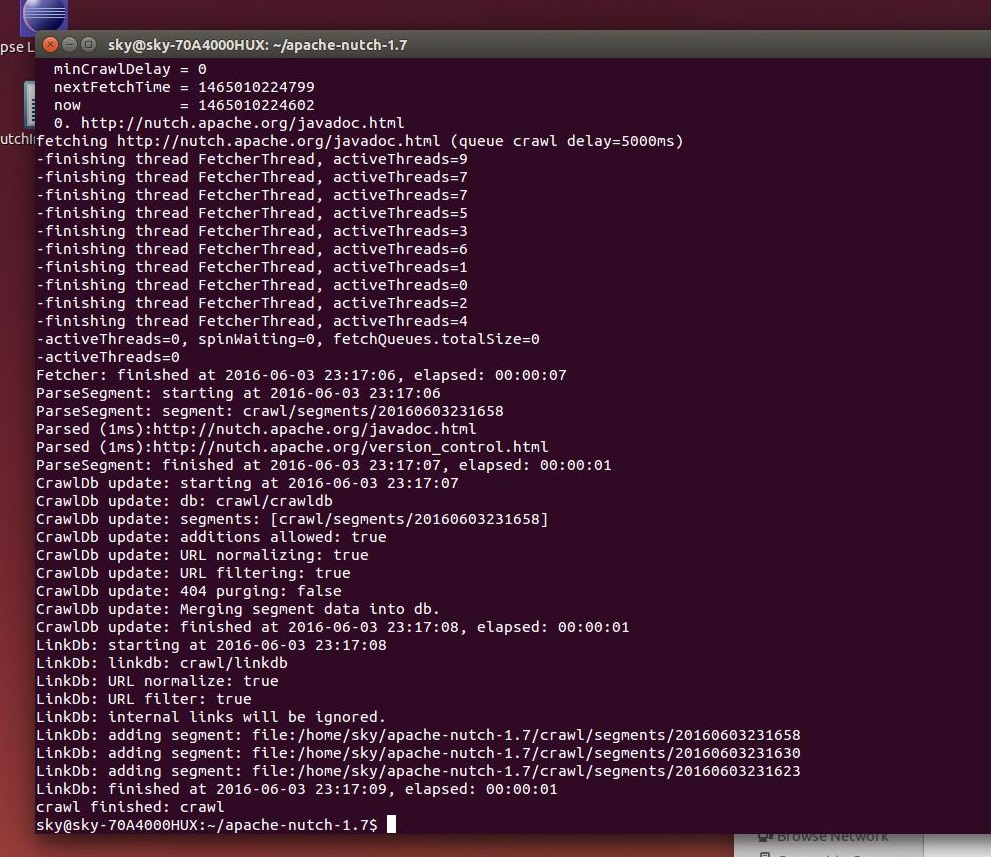
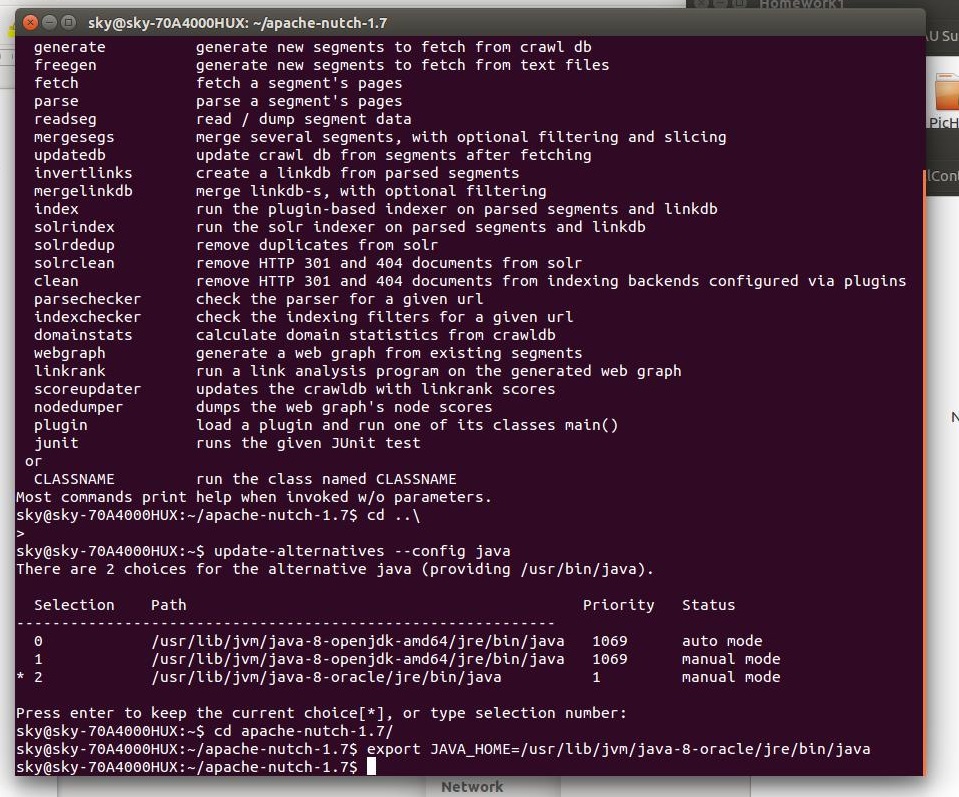
**Part 3:** Hands-on Apache Nutch Web Crawler Project Practice [3.75 pts]

1. Please follow the instructions in the “Apache.Nutch.installation.docx” file, install and configure a Nutch web crawler. Please capture three screenshots to show that
   1. Cygwin has been properly installed and running on your computer [0.5 pt]

* I run nutch on Ubuntu 14.04 => no need to install Cygwin (at first I installed Cygwin on windows 10, I tried many times but all failed, so I ran it on Ubuntu)
  1. Nutch has been downloaded and configured as showing in the instructions [0.5 pt]

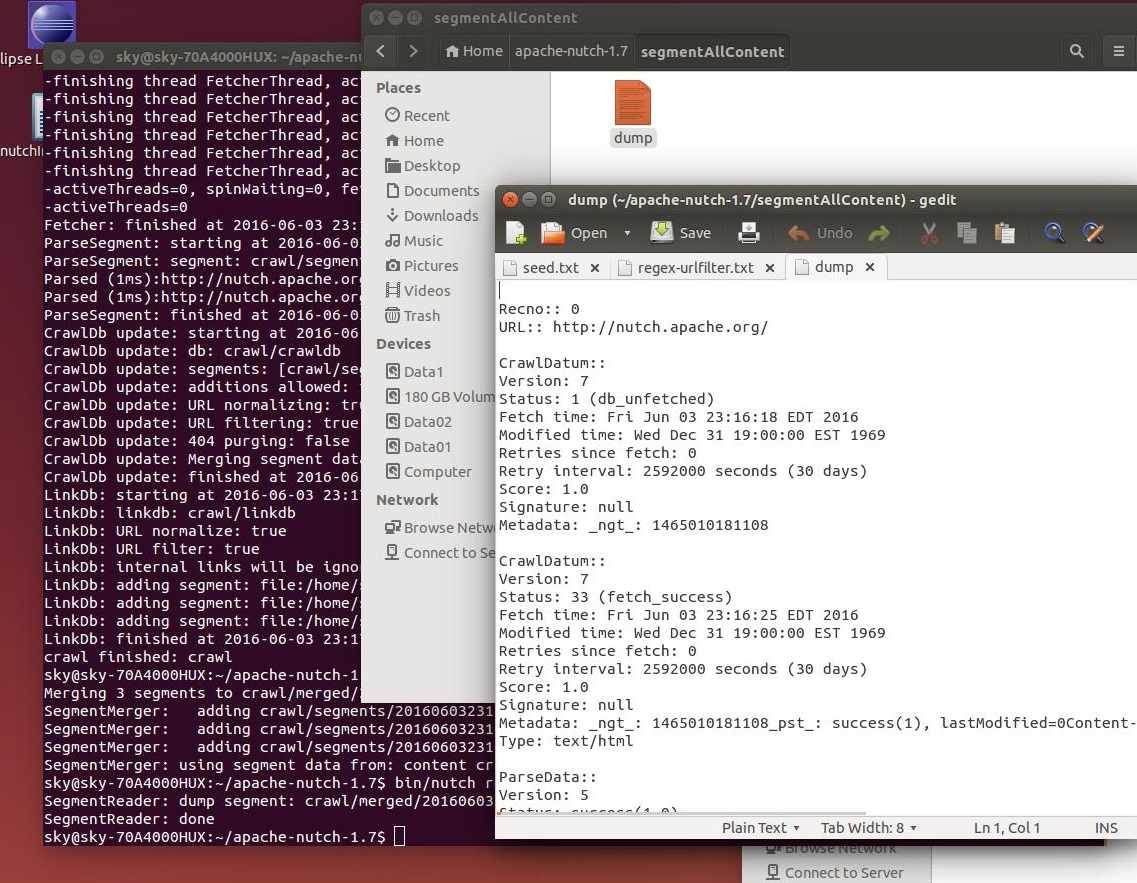
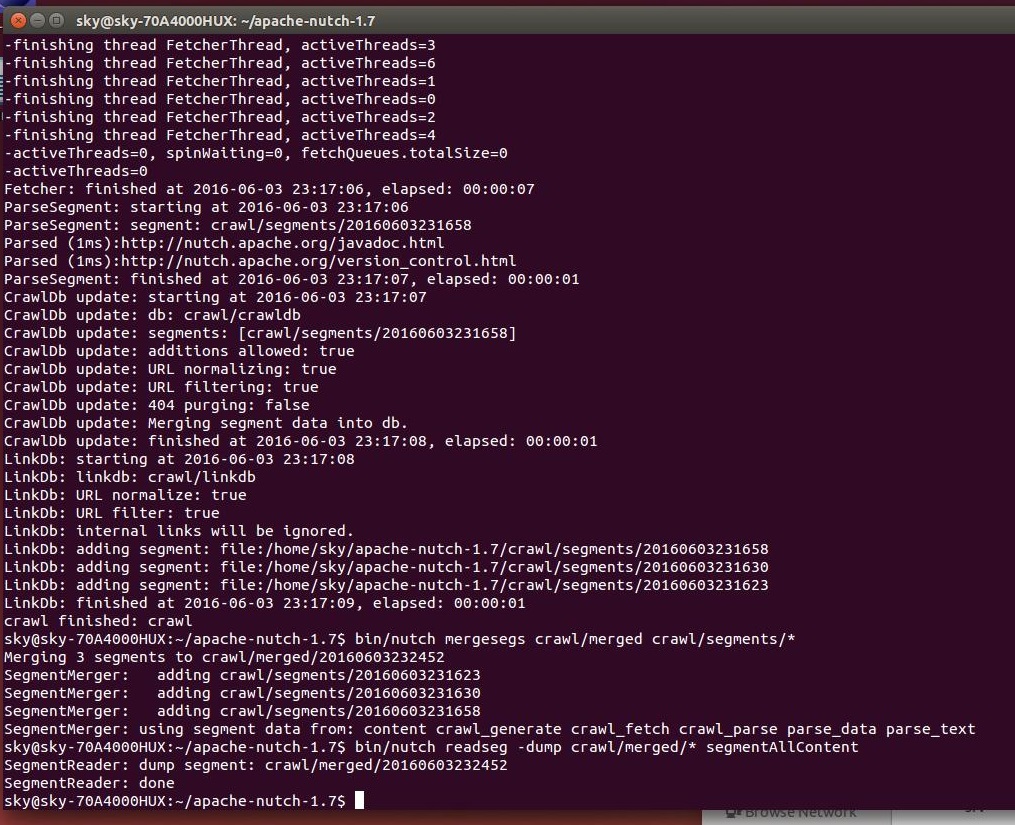


* 1. Ntch can successfully lunch a web crawler task. [1 pt]



1. Please provide a seed URL: I chose <http://www.bbc.com/>

The first time I run *bin/nutch crawl urls -dir crawl* I had depth =300, and topN =100. It took more than 4 hours to finish crawling. Then I changed depth =30 and topN =50, it only took about half an hour to finish the job. See the dump file for the result.



1. Please explain the meaning of the Nutch parameters,

*bin/nutch crawl urls -dir crawl -depth 3 -topN 5*

Meaning of some parameters (these points are from your note)

**-dir** *dir* names the directory to put the crawl in (in this case it is in the urls folder).

**-threads** *threads* determines the number of threads that will fetch in parallel.

**-depth** *depth* indicates the link depth from the root page that should be crawled.

**-topN** *N* determines the maximum number of pages that will be retrieved at each level up to the depth.

As I mentioned in item #2 that when I gave the depth a big number (300) and topN =100, nutch takes a lot of time to crawl the content and the links to other websites.