Nutch tutorial

Table of contents

1. Requirements

1. Java 1.4.x, either from Sun (http://java.sun.com/j2se/downloads.html) or IBM (http://www-106.ibm.com/developerworks/java/jdk/) on Linux is preferred. Set `NUTCH_JAVA_HOME` to the root of your JVM installation.
2. Apache's Tomcat (http://jakarta.apache.org/tomcat/) 4.x.
3. On Win32, cygwin (http://www.cygwin.com/) , for shell support. (If you plan to use Subversion on Win32, be sure to select the subversion package when you install, in the "Devel" category.)
4. Up to a gigabyte of free disk space, a high-speed connection, and an hour or so.

2. Getting Started

First, you need to get a copy of the Nutch code. You can download a release from http://www.nutch.org/release/. Unpack the release and connect to its top-level directory. Or, check out the latest source code from subversion (version_control.html) and build it with Ant (http://ant.apache.org/) .

Try the following command:

```
bin/nutch
```

This will display the documentation for the Nutch command script.

Now we're ready to crawl. There are two approaches to crawling:

1. Intranet crawling, with the `crawl` command.
2. Whole-web crawling, with much greater control, using the lower level `inject`, `generate`, `fetch` and `updatedb` commands.

3. Intranet Crawling

Intranet crawling is more appropriate when you intend to crawl up to around one million pages on a handful of web servers.

3.1. Intranet: Configuration

To configure things for intranet crawling you must:

1. Create a flat file of root urls. For example, to crawl the `nutch.org` site you might start with a file named `urls` containing just the Nutch home page. All other Nutch pages should be reachable from this page. The `urls` file would thus look like:

   ```
   http://www.nutch.org/
   ```
2. Edit the file `conf/crawl-urlfilter.txt` and replace `MY.DOMAIN.NAME` with

the name of the domain you wish to crawl. For example, if you wished to limit the crawl to the `nutch.org` domain, the line should read:

```
+^http://([a-z0-9]*\.)*nutch.org/
```

This will include any url in the domain `nutch.org`.

3.2. Intranet: Running the Crawl

Once things are configured, running the crawl is easy. Just use the crawl command. Its options include:

- `-dir` *dir* names the directory to put the crawl in.
- `-depth` *depth* indicates the link depth from the root page that should be crawled.
- `-delay` *delay* determines the number of seconds between accesses to each host.
- `-threads` *threads* determines the number of threads that will fetch in parallel.

For example, a typical call might be:

```
bin/nutch crawl urls -dir crawl.test -depth 3 >& crawl.log
```

Typically one starts testing one's configuration by crawling at low depths, and watching the output to check that desired pages are found. Once one is more confident of the configuration, then an appropriate depth for a full crawl is around 10.

Once crawling has completed, one can skip to the Searching section below.

4. Whole-web Crawling

Whole-web crawling is designed to handle very large crawls which may take weeks to complete, running on multiple machines.

4.1. Whole-web: Concepts

Nutch data is of two types:

1. The web database. This contains information about every page known to Nutch, and about links between those pages.
2. A set of segments. Each segment is a set of pages that are fetched and indexed as a unit. Segment data consists of the following types:
3. - a *fetchlist* is a file that names a set of pages to be fetched
   - the *fetcher output* is a set of files containing the fetched pages
   - the *index* is a Lucene-format index of the fetcher output.

In the following examples we will keep our web database in a directory named `db` and our segments in a directory named `segments`:

```
mkdir db
mkdir segments
```

4.2. Whole-web: Boostrapping the Web Database

The admin tool is used to create a new, empty database:

```
bin/nutch admin db -create
```

The *injector* adds urls into the database. Let's inject URLs from the DMOZ (http://dmoz.org/) Open Directory. First we must download and uncompress the file listing all of the DMOZ pages. (This is a 200+Mb file, so this will take a few minutes.)

```
wget http://rdf.dmoz.org/rdf/content.rdf.u8.gz
gunzip content.rdf.u8.gz
```

Next we inject a random subset of these pages into the web database. (We use a random subset so that everyone who runs this tutorial doesn't hammer the same sites.) DMOZ contains around three million URLs. We inject one out of every 3000, so that we end up with around 1000 URLs:

```
bin/nutch inject db -dmozfile content.rdf.u8 -subset 3000
```

This also takes a few minutes, as it must parse the full file.

Now we have a web database with around 1000 as-yet unfetched URLs in it.

4.3. Whole-web: Fetching

To fetch, we first generate a fetchlist from the database:

```
bin/nutch generate db segments
```

This generates a fetchlist for all of the pages due to be fetched. The fetchlist is placed in a newly created segment directory. The segment directory is named by the time it's created. We save the name of this segment in the shell variable `s1`:

```
s1=`ls -d segments/2* | tail -1`
echo $s1
```

Now we run the fetcher on this segment with:

```
bin/nutch fetch $s1
```

When this is complete, we update the database with the results of the fetch:

```
bin/nutch updatedb db $s1
```

Now the database has entries for all of the pages referenced by the initial set.

Next we run five iterations of link analysis on the database in order to prioritize which pages to next fetch:

```
bin/nutch analyze db 5
```

Now we fetch a new segment with the top-scoring 1000 pages:

```
bin/nutch generate db segments -topN 1000
s2=`ls -d segments/2* | tail -1`
echo $s2

bin/nutch fetch $s2
bin/nutch updatedb db $s2
bin/nutch analyze db 2
```

Let's fetch one more round:

```

bin/nutch generate db segments -topN 1000
s3=`ls -d segments/2* | tail -1`
echo $s3

bin/nutch fetch $s3
bin/nutch updatedb db $s3
bin/nutch analyze db 2
```

By this point we've fetched a few thousand pages. Let's index them!

4.4. Whole-web: Indexing

To index each segment we use the `index` command, as follows:

```
bin/nutch index $s1
bin/nutch index $s2
bin/nutch index $s3
```

Then, before we can search a set of segments, we need to delete duplicate pages. This is done with:

```
bin/nutch dedup segments dedup.tmp
```

Now we're ready to search!

4.5. Searching

To search you need to put the nutch war file into your servlet container. (If instead of downloading a Nutch release you checked the sources out of CVS, then you'll first need to build the war file, with the command `ant war`.)

Assuming you've unpacked Tomcat as ~/local/tomcat, then the Nutch war file may be installed with the commands:

```
rm -rf ~/local/tomcat/webapps/ROOT*
cp nutch*.war ~/local/tomcat/webapps/ROOT.war
```

The webapp finds its indexes in `./segments`, relative to where you start Tomcat, so, if you've done intranet crawling, connect to your crawl directory, or, if you've done whole-web crawling, don't change directories, and give the command:

```
~/local/tomcat/bin/catalina.sh start
```

Then visit http://localhost:8080/ and have fun!