NutchTutorialPre1.3

Requirements

1. Java 1.4.x, either from Sun or IBM on Linux is preferred. Set NUTCH_JAVA_HOME to the root of your JVM installation. Nutch 0.9 requires Sun JDK 1.5 or higher.
2. Apache's Tomcat 5.x. or higher.
3. On Win32, cygwin, 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.

Getting Started

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

Try the following command:

```
bin/nutch
```

This will display the documentation for the Nutch command script.

Good! You are almost ready to crawl. You need to give your crawler a name. This is required.

1. Edit $NUTCH_HOME/conf/nutch-site.xml and add

```
<property>
  <name>http.agent.name</name>
  <value>YOUR_CRAWLER_NAME_HERE</value>
</property>
```

1. Replace YOUR_CRAWLER_NAME_HERE with the name you want to give to your crawler
2. Optionally you may also set the `http.agent.url` and `http.agent.email` properties so that webmasters can identify who is crawling their site and contact you if necessary.

Note : It is advised to specify your parameters in the file nutch-site.xml and leave nutch-default.xml as it is. The latter should be used as a reference only for checking the list of available parameters and their descriptions.

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

1. Using the **crawl** command to perform all the crawl steps with a single command. This is sometimes referred to as **Intranet Crawling**. Although a simple way to get started, it has limitations.
2. Using the lower level inject, generate, fetch and updatedb commands. Sometimes referred to as **Whole-Web Crawling** this allows you more control of each step of the process and is required to be able to update existing data.

The Crawl Command

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

Crawl Command: Configuration

To configure things for the crawl command you must:

- Create a directory with a flat file of root urls. For example, to crawl the nutch site you might start with a file named urls/nutch containing the url of just the Nutch home page. All other Nutch pages should be reachable from this page. The urls/nutch file would thus contain:

  ```
  http://lucene.apache.org/nutch/
  ```

- Edit the file conf/regex-urlfilter.txt and replace

```
# accept anything else
+.
```

with a regular expression matching the domain you wish to crawl. For example, if you wished to limit the crawl to the apache.org domain, the line should read:

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

This will include any url in the domain apache.org.

Crawl Command: 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.
- **-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.

For example, a typical call might be:

- ```
  bin/nutch crawl urls -dir crawl -depth 3 -topN 50
  ```

Typically one starts testing one's configuration by crawling at shallow depths, sharply limiting the number of pages fetched at each level (-topN), and watching the output to check that desired pages are fetched and undesirable pages are not. Once one is confident of the configuration, then an appropriate depth for a full crawl is around 10. The number of pages per level (-topN) for a full crawl can be from tens of thousands to millions, depending on your resources.

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

Step-by-Step or Whole-web Crawling

Whole-web crawling is designed to handle very large crawls which may take weeks to complete, running on multiple machines. This also permits more control over the crawl process, and incremental crawling. It is important to note that whole web crawling does not necessarily mean crawling the entire world wide web. We can limit a whole web crawl to just a list of the URLs we want to crawl. This is done by using a filter just like we the one we used when we did the crawl command (above).

Step-by-Step: Concepts

Nutch data is composed of:

1. The crawl database, or crawldb. This contains information about every url known to Nutch, including whether it was fetched, and, if so, when.
2. The link database, or linkdb. This contains the list of known links to each url, including both the source url and anchor text of the link.
3. A set of segments. Each segment is a set of urls that are fetched as a unit. Segments are directories with the following subdirectories:
    - a *crawl_generate* names a set of urls to be fetched
    - a *crawl_fetch* contains the status of fetching each url
    - a *content* contains the raw content retrieved from each url
    - a *parse_text* contains the parsed text of each url
    - a *parse_data* contains outlinks and metadata parsed from each url
    - a *crawl_parse* contains the outlink urls, used to update the crawldb
4. The indexes are Lucene-format indexes.

Step-by-Step: Seeding the Crawl DB with a list of URLS

Option 1: Bootstrapping from the DMOZ database.

The injector adds urls to the crawldb. Let's inject URLs from the DMOZ 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 select a random subset of these pages. (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 select one out of every 5000, so that we end up with around 1000 URLs:

```
mkdir dmoz
bin/nutch org.apache.nutch.tools.DmozParser content.rdf.u8 -subset 5000 > dmoz/urls
```

The parser also takes a few minutes, as it must parse the full file. Finally, we initialize the crawl db with the selected urls.
```

```
bin/nutch inject crawl/crawldb dmoz
```

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

Option 2. Bootstrapping from an initial seed list.

Instead of bootstrapping from DMOZ, we can create a text file called `urls`, this file should have one url per line. We can initialize the crawl db with the selected urls.

```
bin/nutch inject crawl/crawldb urls
```

NOTE: version 0.8 and higher requires that we put this file into a subdirectory, e.g. `seed/urls` - in this case the command looks like this:

```
bin/nutch inject crawl/crawldb seed
```

Step-by-Step: Fetching

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

```
bin/nutch generate crawl/crawldb crawl/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 crawl/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 crawl/crawldb $s1
```

Now the database contains both updated entries for all initial pages as well as new entries that correspond to newly discovered pages linked from the initial set.

Now we generate and fetch a new segment containing the top-scoring 1000 pages:

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

bin/nutch fetch $s2
bin/nutch updatedb crawl/crawldb $s2
```

Let's fetch one more round:

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

bin/nutch fetch $s3
bin/nutch updatedb crawl/crawldb $s3
```

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

Step-by-Step: Indexing

Before indexing we first invert all of the links, so that we may index incoming anchor text with the pages.

```
bin/nutch invertlinks crawl/linkdb -dir crawl/segments
```

NOTE: the invertlinks command only applies to Nutch 0.8 and higher.

To index the segments we use the index command, as follows:

```
bin/nutch index crawl/indexes crawl/crawldb crawl/linkdb crawl/segments/*
```

Now we're ready to search!

Command Line Searching

Simplest way to verify the integrity of your crawl is to launch NutchBean from command line:

```
bin/nutch org.apache.nutch.searcher.NutchBean apache
```

where *apache* is the search term (note that NutchBean will only search pages in the `crawl` directory, so if you named the crawl directory something else, NutchBean will not find any results). After you have verified that the above command returns results you can proceed to setting up the web interface.

Installing in Tomcat

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 SVN, 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:

```
mkdir ~/local/tomcat/webapps/nutch
cp nutch*.war ~/local/tomcat/webapps/nutch/
jar xvf ~/local/tomcat/webapps/nutch/nutch-1.1.war
rm nutch-1.1.war;
```

The webapp finds its indexes in ./crawl, relative to where you start Tomcat, so use a command like (platform dependent):

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

If you want to put your search index at a different location. Edit the webapps/nutch/WEB-INF/classes/nutch-site.xml and add the following

```
<property>
<name>searcher.dir</name>
<value>/somewhere/crawl<value> <!-- There must be a crawl/index directory to run off !-->
</property>
```

If your index is changed you need to restart Tomcat with a command like (platform dependent):

```
/etc/init.d/tomcat restart
```

Also it is recommended to make a copy of the index for Tomcat, so that you can crawl and update your index independently.

Then visit: http://localhost:8080/nutch/