OldFAQs

This is the official resource fod OLD Nutch FAQs.

- My system does not find the segments folder. Why? Or: How do I tell the Nutch Servlet where the index file are located?

Fetching
- How many concurrent threads should I use?

Indexing
- Is it possible to change the list of common words without crawling everything again?
- How do I index my local file system?

Searching
- Common words are saturating my search results.
- How is scoring done in Nutch? (Or, explain the "explain" page?)
- How can I influence Nutch scoring?
- What is the RSS symbol in search results all about?

My system does not find the segments folder. Why? Or: How do I tell the *Nutch Servlet* where the index file are located?

There are at least two choices to do that:

- First you need to copy the .WAR file to the servlet container webapps folder.

```
   % cp nutch-0.7.war $CATALINA_HOME/webapps/ROOT.war
```

- 1) After building your first index, start Tomcat from the index folder.
    - Assuming your index is located at /index :

```
% cd /index/
% $CATATALINA_HOME/bin/startup.sh
```

- **Now you can search.**
- 2) After building your first index, start and stop Tomcat which will make Tomcat extrat the Nutch webapp. Than you need to edit the nutch-site.xml and put in it the location of the index folder.

```
% $CATATALINA_HOME/bin/startup.sh
% $CATATALINA_HOME/bin/shutdown.sh
```

```
% vi $CATATALINA_HOME/bin/webapps/ROOT/WEB-INF/classes/nutch-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="nutch-conf.xsl"?>

<nutch-conf>

  <property>
    <name>searcher.dir</name>
    <value>/your_index_folder_path</value>
  </property>

</nutch-conf>

% $CATATALINA_HOME/bin/startup.sh
```

Fetching

How many concurrent threads should I use?

This is dependent on your particular setup, but the following generally works:

If you are using a slow Internet connection (ie- DSL), you might be limited to 40 or fewer concurrent fetches.

If you have a fast Internet connection (> 10Mb/sec) your bottleneck will definitely be in the machine itself (in fact you will need multiple machines to saturate the data pipe). Empirically we find that the machine works well up to about 1000-1500 threads.

To get this to work on my Linux box I needed to set the ulimit to 65535 (ulimit -n 65535), and I had to make sure that the DNS server could handle the load (we had to speak with our colo to get them to shut off an artificial cap on the DNS servers). Also, in order to get the speed up to a reasonable value, we needed to set the maximum fetches per host to 100 (otherwise we get a quick start followed by a very long slow tail of fetching).

To other users: please add to this with your own experiences, my own experience may be atypical.

Indexing

Is it possible to change the list of common words without crawling everything again?

Yes. The list of common words is used only when indexing and searching, and not during other steps. So, if you change the list of common words, there is no need to re-fetch the content, you just need to re-create segment indexes to reflect the changes.

How do I index my local file system?

The tricky thing about Nutch is that out of the box it has most plugins disabled and is tuned for a crawl of a "remote" web server - you **have** to change config files to get it to crawl your local disk.

- 1) crawl-urlfilter.txt needs a change to allow file: URLs while not following http: ones, otherwise it either won't index anything, or it'll jump off your disk onto web sites.
    - Change this line: -(file|ftp|mailto|https): to this: -(http|ftp|mailto|https):
      2) crawl-urlfilter.txt may have rules at the bottom to reject some URLs. If it has this fragment it's probably ok:
    - # accept anything else +.*
      3) By default the protocol-file plugin is disabled. nutch-site.xml needs to be modified to allow this plugin. Add an entry like this:

```
<property>
  <name>plugin.includes</name>
  <value>protocol-file|...copy original values from nutch-default here...</value>
</property>
```

where you should copy and paste all values from nutch-default.xml in the plugin.includes setting provided there. This will ensure that all plug-ins normally enabled will be enabled, plus the protocol-file plugin. Make sure that urlfilter-regex is included, or else **the urlfilter files will be ignored**, leadingNnutch to accept all URLs. You need to enable crawl URL filters to prevent Nutch from crawling up the parent directory, see below.

Now you can invoke the crawler and index all or part of your disk. The only remaining gotcha is that if you use Mozilla it will **not** load file: URLs from a web paged fetched with http, so if you test with the Nutch web container running in Tomcat, annoyingly, as you click on results nothing will happen as Mozilla by default does not load file URLs. This is mentioned here and this behaviour may be disabled by a preference (see security.checkloaduri). IE5 does not have this problem.

Searching

Common words are saturating my search results.

You can tweak your conf/common-terms.utf8 file after creating an index through the following command:

- bin/nutch org.apache.nutch.indexer.HighFreqTerms -count 10 -nofreqs index

How is scoring done in Nutch? (Or, explain the "explain" page?)

Nutch is built on Lucene. To understand Nutch scoring, study how Lucene does it. The formula Lucene uses scoring can be found at the head of the Lucene Similarity class in the Lucene Similarity Javadoc. Roughly, the score for a particular document in a set of query results, "score(q,d)", is the sum of the score for each term of a query ("t in q"). A terms score in a document is itself the sum of the term run against each field that comprises a document ("title" is one field, "url" another. A "document" is a set of "fields"). Per field, the score is the product of the following factors: Its "tf" (term freqency in the document), a score factor "idf" (usually a factor made up of frequency of term relative to amount of docs in index), an index-time boost, a normalization of count of terms found relative to size of document ("lengthNorm"), a similar normalization is done for the term in the query itself ("queryNorm"), and finally, a factor with a weight for how many instances of the total amount of terms a particular document contains. Study the lucene javadoc to get more detail on each of the equation components and how they effect overall score.

Interpreting the Nutch "explain.jsp", you need to have the above cited Lucene scoring equation in mind. First, notice how we move right as we move from "score total", to "score per query term", to "score per query document field" (A document field is not shown if a term was not found in a particular field). Next, studying a particular field scoring, it comprises a query component and then a field component. The query component includes query time – as opposed to index time – boost, an "idf" that is same for the query and field components, and then a "queryNorm". Similar for the field component ("fieldNorm" is an aggregation of certain of the Lucene equation components).

How can I influence Nutch scoring?

Scoring is implemented as a filter plugin, i.e. an implementation of the ScoringFilter class. By default, the OPIC Scoring Filter is used.

However, the easiest way to influence scoring is to change query time boosts (Will require edit of nutch-site.xml and redeploy of the WAR file). Query-time boost by default looks like this:

```
  query.url.boost, 4.0f
  query.anchor.boost, 2.0f
  query.title.boost, 1.5f
  query.host.boost, 2.0f
  query.phrase.boost, 1.0f
```

From the list above, you can see that terms found in a document URL get the highest boost with anchor text next, etc.

Anchor text makes a large contribution to document score (You can see the anchor text for a page by browsing to "explain" then editing the URL to put in place "anchors.jsp" in place of "explain.jsp").

What is the RSS symbol in search results all about?

Clicking on the RSS symbol sends the current query back to Nutch to a servlet named OpenSearchServlet. OpenSearchServlet reruns the query and returns the results formatted instead as RSS (XML). The RSS format is based on OpenSearch RSS 1.0 from a9.com: "OpenSearch RSS 1.0 is an extension to the RSS 2.0 standard, conforming to the guidelines for RSS extensibility as outlined by the RSS 2.0 specification" (See also opensearch). Nutch in turn makes extension to OpenSearch. The Nutch extensions are identified by the 'nutch' namespace prefix and add to OpenSearch navigation information, the original query, and all fields that are available at search result time including the Nutch page boost, the name of the segment the page resides in, etc.

Results as RSS (XML) rather than HTML are easier for programmatic clients to parse: such clients will query against OpenSearchServlet rather than search.jsp. Results as XML can also be transformed using XSL stylesheets, the likely direction of UI development going forward.