Refactoring the Ht://Dig Search Engine

J. Neal Richter
RightNow Technologies
40 Enterprise Blvd.
Bozeman, MT 59718
nealr@rightnow.com

Anthony Arnone
RightNow Technologies
40 Enterprise Blvd.
Bozeman, MT 59718
aarnone@rightnow.com

ABSTRACT

In this paper, we highlight the trials and travails of updating an antiquated and monolithic but full featured web spider and search engine in order to align it more closely with the needs of modern users.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Content Analysis and Indexing—*indexing methods*; H.3.3 [**Information Systems**]: Information search and retrieval—*search process, query formulation*

Keywords

Search, search engines, web spiders, Lucene

1. INTRODUCTION

This is a summary of the effort to bring the well known ht://Dig package up to date. The Hiding 4.0 package has three major parts, the spidering/network module, the indexing storage module, and the query processing module, plus numerous utilities. In addition to removing object spaghetti code and cruft that build up over the years, we replaced the internal storage/query engine and HTML parser as well as converted the package into a usable 'C' shared library.

2. HISTORY AND DESIGN PROBLEMS

In 1995 Andrew Scherpbier began building ht://Dig for use as a search engine and spider at San Diego State University. By the late 90s HtDig had been incorporated into many Linux distributions and was commonly used on many websites. A group of about 20 hobby developers lead by Geoff Hutchison and Gilles Detillieux added to functionality and maintained the software. In 2003 and 2004 RightNow Technologies began utilizing the software for a generic CGI-based search engine. During this time the group became more formal with a set of by-laws and official membership roster, as well as changing the license of HtDig to the LGPL from the GPL. HtDig 3.2 development proceeded and a subproject called libHtDig was developed to enable its use in other software as a shared library.

HtDig 3.2 suffers from a number of design problems, while at the same time being quite flexible for users. Like many other software projects it began to be more spaghetti-like in its structure as featurettes were hacked into place. While implemented in C++, the code base does not use OOP methods or design patterns very well. It's core code was written at a time when many useful components like web-spiders and general purpose IR engines did not exist in easy to integrate forms. The general momentum of the code base often precluded utilization of useful external libraries (networking libraries for example). "If it ain't broke don't fix it" caused a level of code stagnation and became unsustainable.

3. MODERNIZATION

In 2006 we began an effort to reform HtDig by replacing sub-components with emerging standard libraries and further remake HtDig into a usable shared library for the searching and sputtering of documents. The chief advantage of HtDig over projects like Natch is that it is smaller in comparison and simpler to set up as it requires few supporting libraries or environments, nor any advanced Apache/Tomcat needs. This places HtDig in the niche use for website administrators and programmers who do not want to devote lots of effort into implementing a more complex search engine package. It also has the benefit of a decade of users and developers adding a rich set of configuration options. This is not to say that it is superior to packages like Natch or SLR, as both of those packages are more active in comparison as well as far more scalable in documents and search load.

3.1 Replacing DB based index

HtDig originally used Berkeley DB as its storage solution. The major problem with the old implementation was that the inverted index schema used one BDB entry/row per word per document per position. This led to index bloat and long processing times of some utility functions. In addition this schema was not easily extensible to add additional sorted fields for instance.

Rather than recode the index, we chose to migrate to using CLucene and the core storage/query engine. Instead of single word insertion, documents are represented as sets of words in multiple fields. Internally, the index structure is arranged as one entry for each unique word, regardless of how many times or in how many documents it shows up. CLucene allows for easy and extensive customization. Usage of BDB was retained to store the list of documents contained in the CLucene index, allowing straight-forward respidering of the documents.

A special API was written for HtDig that would allow documents to be specified as a hash of field names and asso-

WWW 2007, May 8–12 Banff, Canada
.

ciated text, allowing for flexible addition of document fields. Everything from existing text analyzers to the scoring function code were replaced with a CLucene functions.

3.2 UTF-8 Compliance

HtDig originally used an single-byte text encoding, this made supporting non latin-1 based languages impossible. In addition to auditing the code and reworking all code that assumed "char *" strings, we removed the existing HTML parser and replaced it with HTMLTidy to support parsing of UTF-8 documents. We used the more stable UCS2 'build' of CLucene, so we also imported a set of UTF-8 to UCS2 converters to insert documents. The other major benefit of HTMLTidy is that it gracefully handles malformed HTML/XML and supplies the parsed document in an easily usable DOM-like tree.

4. FEATURE ADDITIONS

Along with the more general improvements to the existing features of HtDig, several new features have been added that provide useful ways to interact with the search index.

4.1 Lucene Queries

CLucene supports the construction of very rich multi-field queries as well as supporting the chaining of filtering steps. HtDig allows a search query to be submitted as a group of optional, required, and forbidden sub-queries. Lucene's support of these features as well as the indexing of text in both stemmed and unstemmed form allowed the removal of many hackish portions of the HtDig codebase that in comparison performed badly.

4.2 Single Document Handling

Single documents can be inserted directly into the searchable index, much like Solr. When inserted, the document can be marked as spiderable or not, in which case it will be downloaded and parsed just as a regular document during the next spider run. If the document is not marked spiderable, it is inserted directly - making it searchable, yet ignored during future spider runs.

Conversely, sometimes it is convenient to spider and store a single document/URL while using the URL rewriting and normalization engine. This would allow a programmer to build a custom spidering engine independent of the internal implementation of HtDig.

4.3 Removed Features

Originally, several different auxiliary databases were available for "fuzzy" searching though the htfuzzy tool. Algorithms like synonyms, metaphone and soundex have been removed completely, the endings database has been subsumed into the main CLucene index as a searchable stemmed field, and the accents database is no longer necessary, being required only in the context of an ASCII-only index. Also removed are some of the database management tools like htmerge, htdump, and htload. The htmerge tool was used to merge BDB indexes created during different runs of htdig into one.

Gone also is the CGI based searching executable that was part of ht://Dig. This has been replaced with a simple PHP API that provides the same functionality. This conforms better to the idea of using HtDig as a utility library.

5. FUTURE DIRECTIONS

HtDig is still a work in progress. Many features have yet to be implemented, and some of the previously removed features could stand to be replaced with modern analogues from the rich set of advanced Lucene code. This will help keep HtDig useful in its intended niche.

5.1 Tool Collection

Some of the tools removed from ht://Dig should be reincorporated back into the program, as they provide useful utilities. Some possibilities are htmerge and htdump. These could be implemented as a nice front end GUI for database maintenance. Features could include things like automatic URL blacklisting and manual field manipulation. Note that tools like Luke allow examination of a Lucene index, yet they do not perform all useful index editing tasks.

5.2 External APIs

Continuing in the vein of standardization, HtDig's external APIs should be updated even further to a well known, standardized specification. A standardized RSS based API like A9's OpenSearch provides a good example for an accessible search utility.

Also, an XML based configuration feature would allow for easier and more accessible customization of the spidering process, along with streamlining communications for programs that incorporate HtDig as a utility library.

5.3 Usage with Solr

One of the most useful portions of HtDig code is the battle tested spidering and network code. One potential idea is to instrument the spider to feed a Solr-based servlet engine with Documents via XML. Solr in many respects is more powerful that using Lucene alone, yet still suffers from the drawback that it has no tightly integrated spider.

5.4 Usage with Nutch

Nutch is a very active daemon based search and spidering engine that supports high scalability. One area of interest to us would be to implement a nearly identical Lucene schema to Nutch. This could potentially allow a user an upgrade path to a more scalable solution without necessitating the rebuilding of their entire index. HtDig users that have become comfortable and familiar with the configuration of the HtDig and its spider would still be able to use this knowledge while getting the benefits of a high capacity engine like Nutch.

6. ACKNOWLEDGMENTS

We would like to acknowledge Ben van Klinken, for his work on CLucene as well as Doug Cutting for his work on Lucene. It also can not be understated how much work that Gilles and Geoff did for HtDig's code and users for many years before moving on to other pursuits.

7. LINKS

HtDig Project homepage: http://www.htdig.org/
Lucene Project homepage: http://lucene.apache.org/
CLucene Project homepage: http://clucene.sourceforge.net/
HTMLTidy Project homepage: http://tidy.sourceforge.net/
Solr Project homepage: http://lucene.apache.org/solr/
Nutch Project homepage: http://lucene.apache.org/nutch/