NUTCH CHEAT SHEET

What is NUTCH you ask?

Nutch is a very popular open source JAVA based search engine built on top of Lucene which is translated to C, C++, C#, Python, Perl and Ruby. It provides all of the tools you need to run your very own search engine. Current version of nutch (as of October 2010) is: 1.2

Some important gotchas on nutch:

- Founded in 2003 by Doug Cutting, the Lucene creator, and Mike Cafarella
- Apache project since 2004 (sub-project of Lucene)
- Spin-offs:
  - Map-Reduce and distributed FS → Hadoop
  - Content type detection and parsing → Tika
- Many installations in operation, mostly vertical search
- Collections typically 1 mln - 200 mln documents

Usage:

nutch [-core] COMMAND

where COMMAND is one of the commands from following table or CLASSNAME the class named CLASSNAME most commands print help when invoked w/o parameters.

Expert: -core option is for developers only. It avoids building the job jar, instead it simply includes classes compiled with ant compile-core. [NOTE: this works only for jobs executed in 'local' mode]

Command	Description	Usage	Example
crawl	one-step crawler for intranets	Crawl <urlDir> [-dir d] [-threads n] [-depth i] [-topN N] [-solr solrURL]	bin/nutch crawl urls -dir crawl -depth 3 -topN 50
readdb	read / dump crawl db	CrawlDbReader <crawldb> (-stats | -dump <out_dir> | -topN <nnnn> <out_dir> [<min>] | -url <url>) <crawldb> directory name where crawldb is located -stats [-sort] print overall statistics to System.out [-sort] list status sorted by host -dump <out_dir> [-format normal|csv] dump the whole db to a text file in <out_dir> [-format csv] dump in Csv format	1. bin/nutch readdb crawl/crawldb -dump reports/dg 2. bin/nutch readdb crawl/crawldb -stats

		[-format normal] dump in standard format (default option) -url <url> print information on <url> to System.out -topN <nnnn> <out_dir> [<min>] dump top <nnnn> urls sorted by score to <out_dir> [<min>] skip records with scores below this value. This can significantly improve performance.	
convdb	convert crawl db from pre-0.9 format	CrawlDbConverter <oldDb> <newDb> [-withMetadata] oldDb name of the crawldb that uses UTF8 class. newDb name of the output crawldb that will use Text class. withMetadata convert also all metadata keys that use UTF8 to Text.	bin/nutch convdb crawl/crawldb
mergedb	merge crawldb-s, with optional filtering	CrawlDbMerger <output_crawldb> <crawldb1> [<crawldb2> <crawldb3> ...] [-normalize] [-filter] output_crawldb output CrawlDb crawldb1 ... input CrawlDb-s (single input CrawlDb is ok) -normalize use URLNormalizer on urls in the crawldb(s) (usually not needed) -filter use URLFilters on urls in the crawldb(s)	bin/nutch mergesegs crawl/segments -dir crawl/segments
readlinkdb	read / dump link db	LinkDbReader <linkdb> {-dump <out_dir> | -url <url>) -dump <out_dir> dump whole link db to a text file in <out_dir> -url <url> print information about <url> to System.out	bin/nutch readlinkdb crawl/linkdb -dump reports/link
inject	inject new urls into the database	Injector <crawldb> <url_dir>	bin/nutch inject crawl/crawldb seed/dynamicguy/url
generate	generate new segments to fetch from crawl db	Generator <crawldb> <segments_dir> [-force] [-topN N] [-numFetchers numFetchers] [-adddays numDays] [-noFilter] [-noNorm][-maxNumSegments num]	1. bin/nutch generate crawl/crawldb crawl/segments 2. bin/nutch generate crawl/crawldb crawl/segm

			ents -topN 100
freegen	generate new segments to fetch from text files	FreeGenerator <inputDir> <segmentsDir> [-filter] [-normalize] inputDir input directory containing one or more input files. Each text file contains a list of URLs, one URL per line segmentsDir output directory, where new segment will be created -filter run current URLFilters on input URLs -normalize run current URLNormalizers on input URLs	bin/nutch freegen -dir seed/dynamicguy/ crawl/segments/*
fetch	fetch a segment's pages	Fetcher [-threads n] [-noParsing]	bin/nutch fetch crawl/segments/20100929225609
parse	parse a segment's pages	ParseSegment segment	bin/nutch parse crawl/segments/20100930105013/
readseg	read / dump segment data	SegmentReader (-dump ... | -list ... | -get ...) [general options] * General options: -nocontent ignore content directory -nofetch ignore crawl_fetch directory -nogenerate ignore crawl_generate directory -noparse ignore crawl_parse directory -noparsedata ignore parse_data directory -noparsetext ignore parse_text directory * SegmentReader -dump <segment_dir> <output> [general options] Dumps content of a <segment_dir> as a text file to <output>. <segment_dir> name of the segment directory. <output> name of the (non-existent) output directory. * SegmentReader -list (<segment_dir1> ... | -dir <segments>) [general options] List a synopsis of segments in specified directories, or all segments in	bin/nutch readseg -list crawl/segments/20100930105013/

		a directory <segments>, and print it on System.out <segment_dir1> ... list of segment directories to process -dir <segments> directory that contains multiple segments * SegmentReader -get <segment_dir> <keyValue> [general options] Get a specified record from a segment, and print it on System.out. <segment_dir> name of the segment directory. <keyValue> value of the key (url). Note: put double-quotes around strings with spaces.	
mergesegs	merge several segments, with optional filtering and slicing	SegmentMerger output_dir (-dir segments | seg1 seg2 ...) [-filter] [-slice NNNN] output_dir name of the parent dir for output segment slice(s) -dir segments parent dir containing several segments seg1 seg2 ... list of segment dirs -filter filter out URL-s prohibited by current URLFilters -slice NNNN create many output segments, each containing NNNN URLs	bin/nutch mergesegs crawl/segments -i -ds
updatedb	update crawl db from segments after fetching	CrawlDb <crawldb> (-dir <segments> | <seg1> <seg2> ...) [-force] [-normalize] [-filter] [-noAdditions] crawldb CrawlDb to update -dir segments parent directory containing all segments to update from seg1 seg2 ... list of segment names to update from -force force update even if CrawlDb appears to be locked (CAUTION advised) -normalize use URLNormalizer on urls in CrawlDb and segment (usually not needed) -filter use URLFilters on urls in CrawlDb and segment -noAdditions only update already existing URLs, don't add any newly discovered URLs	bin/nutch updatedb crawl/crawldb $s2
invertlinks	create a linkdb from parsed segments	CrawlDb <crawldb> (-dir <segments> | <seg1> <seg2> ...) [-force] [-normalize] [-filter] [-noAdditions] crawldb CrawlDb to update -dir segments parent directory containing all segments to update from	bin/nutch invertlinks crawl/linkdb crawl/segments/20100

		seg1 seg2 ... list of segment names to update from -force force update even if CrawlDb appears to be locked (CAUTION advised) -normalize use URLNormalizer on urls in CrawlDb and segment (usually not needed) -filter use URLFilters on urls in CrawlDb and segment -noAdditions only update already existing URLs, don't add any newly discovered URLs	930105013
mergelinkdb	merge linkdb-s, with optional filtering	LinkDbMerger <output_linkdb> <linkdb1> [<linkdb2> <linkdb3> ...] [-normalize] [-filter] output_linkdb output LinkDb linkdb1 ... input LinkDb-s (single input LinkDb is ok) -normalize use URLNormalizer on both fromUrls and toUrls in linkdb(s) (usually not needed) -filter use URLFilters on both fromUrls and toUrls in linkdb(s)	bin/nutch mergelinkdb crawl/indexes/ crawl/crawldb/ crawl/linkdb/ crawl/segments/20100930105013/
index	run the indexer on parsed segments and linkdb	Indexer <index> <crawldb> <linkdb> ...	bin/nutch index crawl/indexes crawl/crawldb crawl/linkdb crawl/segments/*
solrindex	run the solr indexer on parsed segments and linkdb	SolrIndexer <solr url> <crawldb> <linkdb> ...	bin/nutch solrindex http://127.0.0.1:8080/solr/ crawl/crawldb crawl/linkdb crawl/segments/*
merge	merge several segment indexes	IndexMerger [-workingdir <workingdir>] outputIndex indexesDir...	`bin/nutch solrdedup` http://localhost:8983/solr/

dedup	remove duplicates from a set of segment indexes	DeleteDuplicates <indexes> ...	bin/nutch dedup crawl/indexes
solrdedup	remove duplicates from solr	SolrDeleteDuplicates <solr url>	bin/nutch solrdedup http://localhost:8983/solr
plugin	load a plugin and run one of its classes main()	PluginRepository pluginId className [arg1 arg2 ...]	in/nutch plugin parse-html org.apache.nutch.parse.html.HtmlParser path/to/file.html
server	run a search server	DistributedSearch$Server <port> <crawl dir>	bin/nutch server 1234 crawl

Nutch Documents:

Field	Stored	Indexed	analyzed
Url	YES	YES	YES
anchor	NO	YES	YES
content	NO	YES	YES
Site	YES	YES	NO
Lang	YES	YES	NO