NewScoring

This page describes the new scoring (i.e. WebGraph and Link Analysis) functionality available in Nutch 1.X as of revision 723441. See also the new scoring example.

General Information

The new scoring functionality can be found in org.apache.nutch.scoring.webgraph. This package contains multiple programs that build web graphs, perform a stable convergent link-analysis, and update the crawldb with those scores. These programs assume that fetching cycles have already been completed and now the users want to build a global webgraph from those segments and from that webgraph perform link-analysis to get a single global relevancy score for each url. Building a webgraph assumes that all links are stored in the current segments to be processed. Links are not held over from one processing cycle to another. Global link-analysis scores are based on the current links available and scores will change as the link structure of the webgraph changes.

Currently the scoring jobs are not integrated into the Nutch script as commands and must be run in the form bin/nutch org.apache.nutch.scoring.webgraph.XXXX.

WebGraph

The WebGraph program is the first job that must be run once all segments are fetched and ready to be processed. WebGraph is found at org.apache.nutch.scoring.webgraph.WebGraph. Below is a printout of the programs usage.

```
usage: WebGraph
 -help                      show this help message
 -segment <segment>         the segment(s) to use
 -webgraphdb <webgraphdb>   the web graph database to use
```

The WebGraph program can take multiple segments to process and requires an output directory in which to place the completed web graph components. The WebGraph creates three different components: an inlink database, an outlink database, and a node database. The inlink database is a listing of url and all of its inlinks. The outlink database is a listing of url and all of its outlinks. The node database is a listing of url with node meta information including the number of inlinks and outlinks, and eventually the score for that node.

Loops

Once the web graph is built we can begin the process of link analysis. Loops is an optional program that attempts to help weed out spam sites by determining link cycles in a web graph. An example of a link cycle would be sites A, B, C, and D, where A links to B which links to C which links to D which links back to A. This program is computationally expensive and usually, due to time and space requirement, can't be run on more than a three or four level depth. While it does identify sites which appear to be spam and those links are then discounted in the later LinkRank program, its benefit to cost ratio is very low. It is included in this package for completeness and because there may be a better way to perform this function with a different algorithm. But on current large production webgraphs, its use is discouraged. Loops is found at org.apache.nutch.scoring.webgraph.Loops. Below is a printout of the programs usage.

```
usage: Loops
 -help                      show this help message
 -webgraphdb <webgraphdb>   the web graph database to use
```

LinkRank

With the web graph built we can now run LinkRank to perform an iterative link analysis. LinkRank is a PageRank-like link analysis program that converges to stable global scores for each url. Similar to PageRank, the LinkRank program starts with a common score for all urls. It then creates a global score for each url based on the number of incoming links and the scores for those links and the number of outgoing links from the page. The process is iterative and scores tend to converge after a given number of iterations. It is different from PageRank in that nepotistic links such as links internal to a website and reciprocal links between websites can be ignored. The number of iterations can also be configured; by default 10 iterations are performed. Unlike the previous OPIC scoring, the LinkRank program does not keep scores from one processing time to another. The web graph and the link scores are recreated at each processing run and so we don't have the problems of ever increasing scores. LinkRank requires the WebGraph program to have completed successfully and it stores its output scores for each url in the node database of the webgraph. LinkRank is found at org.apache.nutch.scoring.webgraph.LinkRank. Below is a printout of the programs usage.

```
usage: LinkRank
 -help                       show this help message
 -webgraphdb <webgraphdb>    the web graph db to use
```

ScoreUpdater

Once the LinkRank program has been run and link analysis is completed, the scores must be updated into the crawl database to work with the current Nutch functionality. The ScoreUpdater program takes the scores stored in the node database of the webgraph and updates them into the crawldb. If a url exists in the crawldb that doesn't exist in the webgraph then its score is cleared in the crawldb. The ScoreUpdater requires that the WebGraph and LinkRank programs have both been run and requires a crawl database to update. ScoreUpdater is found at org.apache.nutch.scoring.webgraph. ScoreUpdater. Below is a printout of the programs usage.

```
usage: ScoreUpdater
 -crawldb <crawldb>          the crawldb to use
 -help                       show this help message
 -webgraphdb <webgraphdb>    the webgraphdb to use
```

Questions

If internal links are not ignored, would the LinkRank scores be equivalent to PageRank scores?

To understand this we are required to explain how the LinkRank scores are calculated exactly.

The WebGraph and LinkRank classes work together. The WebGraph is where links from either the same domain or same host can be ignored (or allowed). The configuration parameters:

```
link.ignore.internal.host = true|false
link.ignore.internal.domain = true|false
```

can be used to change that behavior. By default it ignores links from the same domain and host. So a link from news.google.com wouldn't be counted and wouldn't raise the score for www.google.com. The WebGraph just builds the lists of inlinks, outlinks, and nodes then the LinkRank class processes that to create the score. LinkRank does follow very closely to the original pagerank formula which is something like:

(1 - dampingFactor) + (dampingFactor * totalInlinkScore)

Where totalInlinkScore is the calculated from all the inlinks pointing to a page, taking into account that this is iterative and pages all start off with rankOne score which is (1 / numLinksInWebGraph).

The differences are:

1. The Loops class can be used to identify and remove spam/problem links. This class was supposed to identify reciprocal links and link cycles and then allow those links to be removed. Problem is the class is very expensive computationally. You can set the depth you want it to run but it is worse than exponential so I wouldn't do more than 1-3 depth if at all. That will get you reciprocal links and small link cycles (a->b->c->a). Really this doesn't add much to score in the end, I would just leave it off and not run this job. 2. You can limit duplicate links from pages and domains. Say page A points to B twice, you can limit it and only count it once. 3. There is a damping factor which is by default set to 0.85. This is the same as the original pagerank paper. This is configurable with the link.analyze.damping.factor parameter. 4. LinkRank runs a given number of iterations. Ideally the job would iterate until the scores converge to a point, currently it is a set number of iterations.

LinkRank scores should be equivalent (close enough) to pagerank scores. Some things to consider:

1. Pagerank is just one of over 200 signals that google uses (if they still use it) to determine relevancy. Even if Google still uses it it most likely has changed. Link analysis scores are good global relevancy scores, but a link score does not a search engine make today. Oh how I wish it was that simple. LinkRank is a good starting point, that's it. 2. This is only as good as the amount of pages you have crawled. The larger your set of crawled segments the better the scores get. 3. A link is a link, it is content agnostic. If you crawl 100m pages and do a LinkRank on that you will see all the usual suspects (Google, YouTube, Facebook) but you will also see things like the flash download. To LinkRank a link is a link, it isn't particular in it being a viewable piece of content.

For more on this topic please see the NewScoringIndexingExample