NutchScoring

Nutch Scoring

This page is dedicated to Scoring implementations within Apache Nutch. The language used within this document is intended to reflect that used within the Nutch community and vocabulary may vary from time to time, words may be used interchangeably to refer to the same thing, etc. If you feel there is a discrepancy with this document then please let us know by contacting us.

Introduction

Amongst other things Apache Nutch is pluggable and modular with extensible interfaces. Parse, Index and ScoringFilter can all use custom implementations. This document explains the basics of scoring in Apache Nutch, including information on:

- What Scoring is... what it means in Nutch.
- Where Scoring takes place within the Nutch Crawl cycle.
- Nutch Scoring extension points and how we can implement custom scoring algorithms.
- Scoring issues currently under development.

What Scoring is... what it means in Nutch

- Describe CrawlDatum data structure in Nutch trunk

A scoring filter will manipulate scoring variables in CrawlDatum and in resulting search indexes. Filters can be chained in a specific order, to provide multi-stage scoring adjustments.

- ./src/java/org/apache/nutch/scoring/ScoringFilter.java - A scoring filter will manipulate scoring variables in CrawlDatum and in resulting search indexes. Filters can be chained in a specific order, to provide multi-stage scoring adjustments.
- ./src/java/org/apache/nutch/scoring/ScoringFilterException.java
- ./src/java/org/apache/nutch/scoring/ScoringFilters.java - Create and cache ScoringFilter implementing plugins.

Where Scoring takes place within the Nutch Crawl cycle

Scoring occurs in numerous places throughout the Nutch codebase and consequently within the crawl cycle. This section describes the point of occurence and functional purpose Scoring serves at each step. You will see that the list of elements has been structured to represent the logical and typical progression of a Nutch crawl cycle.

- ./src/java/org/apache/nutch/crawl/Injector.java - Scoring filters are defined within the various MapReduce job configurations. This means that the desired configuration will be used appropriately at runtime when the job is run by the JobClient. The Injector actually contains two MapReduce jobs, namely
  - sortJob - where we set the InjectMapper as the Mapreduce Mapper override. The InjectMapper uses ScoringFilters to calculate a new initial score for a particular URL based on passing in the Hadoop Text key (representing the URL of the page) and associated CrawlDatum value (representing a new datum for which filters will modify it in-place) to the ScoringFilters.injectedScore method. Essentially this sets an initial score for newly injected pages. It should be noted that newly injected pages may have no inlinks, so filter implementations may wish to set this score to a non-zero value, to give newly injected pages some initial credit. We are concerned with the value for `db.score.injected` in this case as this assigns a default of `1.0f` against the score of new pages added by the injector. This default score can however be overridden by associating the `nutch.score` metadata flag against any URL in a seed list. This allows to set a custom score for a specific URL. If this is the case we assign this score to the CrawlDatum object, if not then we use the default score as described above.
  - mergeJob - which combines multiple new entries for a given URL. An example of when this is necessary would be if we attempt to inject two indentical URLs within the same seed list and where these should be merged into one record. In this job we are concerned with discovering the value for the `db.score.injected` configuration property present within `nutch-site.xml` as populated in the initial sortJob execution as described above. This value represents the score of new pages added by the Injector. This is relevant for us as we must know if a record already exists and we wish to update but not overwrite the value.
- ./src/java/org/apache/nutch/crawl/Generator.java - ScoringFilters are used within the Generator.Selector class selects URL entries due for Fetching. In addition to specifying the ScoringFilters within the MapReduce job configuration, ScoringFilter is used within the Map phase of this job which selects and inverts a subset of URLs due for fetching. In particular we implement the `Generator.Selector.generatorSortValue` method which prepares a sort value for sorting and selecting the top N scoring pages during fetchlist generation. The arguments for Hadoop are: Text key `url` representing the url of the page we are trying to score, CrawlDatum `datum` which represents the page's datum which should not be modified in this task, and an initial sort value `initSort` of `1.0f`. It should be noted that the final value doesn't always need to be set to `1.0f` as it can be linked to a value from previous filters in the chain of Scoring implementations. The result of executing the `Generator.Selector.generatorSortValue` function is subsequently used to consider only entries with a score superior to the threshold. Only URL entries above the threshold will then be fetched.
- ./src/java/org/apache/nutch/fetcher/Fetcher.java -

- ./src/java/org/apache/nutch/crawl/CrawlDbReducer.java
- ./src/java/org/apache/nutch/indexer/IndexerMapReduce.java
- ./src/java/org/apache/nutch/parse/ParseOutputFormat.java
- ./src/java/org/apache/nutch/parse/ParserChecker.java
- ./src/java/org/apache/nutch/parse/ParseSegment.java
- ./src/java/org/apache/nutch/tools/arc/ArcSegmentCreator.java
- ./src/java/org/apache/nutch/tools/FreeGenerator.java

Scoring extension points

Examples

- NewScoring – New stable pagerank like webgraph and link-analysis jobs.
- NewScoringIndexingExample – Two full fetch cycles of commands using new scoring and indexing systems.
- AbstractScoringFilter
- DepthScoringFilter
- LinkAnalysisScoringFilter
- OPICScoringFilter
- TLDScoringFilter
- URLMetaScoringFilter
- SimilarityScoringFilter

Development Issues

FixingOpicScoring - *In planning*.