Voyager

Voyager: Setting Relevancy With Federated Search

June 2013

www.voyagersearch.com

www.voyagersearch.com/demo

Relevancy

Voyager is built using the robust open source search engine Apache Lucene. Relevance in Lucene is based on the classic TF/IDF model (Term Frequency / Inverse Document Frequency). In this model, relevancy is calculated by comparing how often query terms show up in the document (DF), but weighing the results by how often the terms appear in the entire corpus (TF). The result is that documents include words that are rare within to the entire collection are given more weight in related queries. For a deeper look at TF/IDF, see:

http://en.wikipedia.org/wiki/Tf%E2%80%93idf

In addition to standard text relevancy, Voyager exposes filters that promote categories of data within search results. This approach helps refine searches to the most relevant results.

Within the Voyager UI, an administrator can tune how we calculate the relevance ranking. There are three fundamental types of tuning that are all based on SOLR/Lucene.

Query fields - this dictates weighting for keywords. It gives you control for which fields of the index have a higher weight than others when evaluating keyword search. For instance metadata fields might take precedence over other fields such as filename

Default Sort - when results are returned you might want to sort results based on certain fields. For example items from one database server might be sorted over another server and so on..

Append Parameters - use this to boost items. If the users query returns two documents with equal relevancy boost the one that has metadata to the top. Or if you have the same file copied in two different places - return the one that is on a reliable file server over the one that might be in a lesser used location. This option gives admistrators access to the full power of Apache Lucene relevancy.

Federated Search and Relevancy

When searching across multiple systems, relevancy is more complicated because there is no shared corpus on which to base calculations. Some federated search systems simply query each server and get the top few results from each and show merge them together. Without a shared scoring system, the cross-system relevancy is opaque.

Voyager supports federated search by connecting to multiple Voyager instances, and using the same relevancy settings across all servers. Because there is a shared strategy and Voyager can compute aggregate filters across all servers, this produces reasonable results for federated search relevancy.

Alternative Approach

Voyager is able to give reasonable federated relevancy because the various systems share common scoring approaches. While this approach works well for data that fits nicely into Voyager, it does not work when the data cannot be indexed. One possible solution is to build federated cluster search. In this approach a central server collects the top query results from many heterogeneous systems and then scores by querying the corpus of top search results. This approach will improve federated relevancy although it will increase the query times. For an example of this approach, see:

http://search.carrotsearch.com/carrot2-webapp/search