Samuel Martin, programmer
Ravi Shankar, instructor

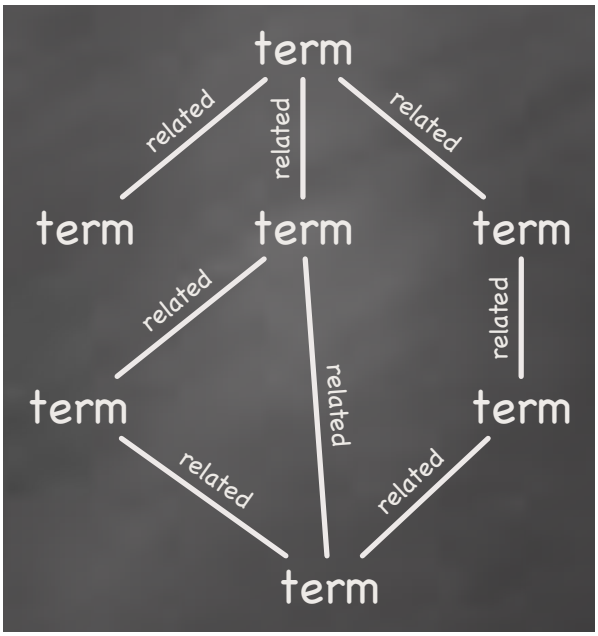
REPORT NAME

Abstract

This project implements Lucene to search an index, but the main focus is on the method used to construct the query. It involves an OWL (Web Ontology Language) ontology containing numerous keywords and their interrelations, and uses that to build a search query with additional, prioritized search terms. This will result in more relevant search results that go in depth about the desired subject.

Design and conceptual development

I didn't do much design before working on this project. I just had the idea and went from there. Most of the design work was done in my head or on the fly. The only thing I really did beforehand was drawing a diagram similar to the following to explain the ontology: Each term included is connected to other related terms.



The algorithm takes a number of user-selected terms constructs a query containing them, then looks for terms related to those, and adds them to the query with decreasing relevance.

Methods

Nutch and Solr were used to crawl and index websites for testing purposes. The Eclipse IDE was used to write the program, as that is the tool of choice for Java development. Protégé-OWL, from Stanford, was used to construct the ontology containing the search terms. Java APIs used include Jena, for reading the ontology file, and Lucene, to read and search the indexed files.

I wrote a Java program to test this concept. It's a fairly basic command-line program, as Java is not my forte. First I created an ontology of keywords centered around the Semantic Web, as it was the first subject to come to mind. Then I generated a Lucene index by crawling a website with Nutch and indexing it with Solr, then wrote the program to test my query algorithm.

The algorithm takes the selected keywords, adds them to a Lucene query, and boosts them with a relatively high value, indicating that they are considered the most important search terms. It then inspects the ontology to find any terms related to the selected terms, excluding the selected terms themselves, and adds those to the query with

a slightly lesser boost value. Then it finds terms related to the ones just added, excluding terms already in the query, and adds them with a lower boost value, and so on.

Results

I have no screenshots to include since my program has no GUI, being a command-line program, so I'll just provide a sample result. This algorithm, with the boost values I have chosen to use, takes a query like:

> RDFa Nutch

and turns it into:

> nutch^4.0 rdfa^4.0 solr^3.0 lucene^3.0 hadoop^3.0 microformats^3.0 "world wide web consortium"^3.0 w3c^3.0 "semantic web"^3.0 "web 3.0"^3.0 "resource description framework"^3.0 rdf^3.0 xml "extensible markup language" "extensible hypertext markup language" xhtml "rdf s" "rdf schema" rdfs "rdf xml" owl "web ontology language" html5 html ontology sparql jena

The "content:" preceding each term has been omitted for brevity.

Discussion

The biggest problem I had in this project was figuring out why Lucene wasn't able to search the index generated by Solr. Eventually, I tried editing a Porter stemming filter out of Solr's schema file, and suddenly my queries worked.

My second biggest problem, and one that I didn't resolve, was coming up with a healthy set of search terms. An expert in a field would be much better equipped to populate an ontology for this. An expert in Lucene would likewise be much better equipped to put these queries to use.

Conclusion

I believe that this algorithm would improve search results for any terms that might be included in an accompanying ontology, particularly if said ontology is populated by someone with a lot of knowledge and experience in the subject.

REFERENCES

Protégé-OWL, from Stanford University, used to create the ontology file.

Jena, an open-source Sourceforge project started by HP Labs, used in the program.

Apache Lucene, used in the program and the focus of the project.

Apache Nutch, used to crawl the web for sample data.

Apache Solr, used to index sample data.