Information Retrieval (IR 6900) Project

Due on Dec 13 (Fri) by 5:00pm

Option 1: OU Web Search Engine, From Scratch

For this project, you will implement and evaluate a full-fledged search engine for the `ohio.edu` domain. You are encouraged to use the code that you have implemented already for the homework assignments. The search engine implementation will need to contain the following components:

1. **Crawler**: use the breadth-first strategy to crawl at least 10K pages. Make sure that you consider `ohiou.edu` pages as in-domain pages.

2. **Index**: use the pre-processing steps discussed in class.

3. **Query Processing**: Implement the following models and offer them as options on the web page of your search engine:

   - **TF.IDF**: use the standard *tf.idf* weighting scheme *ltc.ltn* in the Vector Space Model approach to IR.
   - **BM25**: use the Okapi BM25 extension to BIM, with parameters $k_1 = 1.5$, $k_2 = 1.5$, and $b = 0.754$.
   - **LM**: use a language model with Dirichlet smoothing, with parameter $\alpha = 2,000$.

4. **Web Interface**: Implement the interface through a web page that will be opened in a client browser and an HTTP server that will answer the search requests from the client. Implement the "more results" feature discussed in class.

Furthermore, implement *at least one* of the following extensions:

1. Phrase search.

2. Operators for mandatory terms (+) and negated terms (–).

3. Relevance feedback.

4. Use the title text separately from the rest of the document. The similarity between the title text and the query should be combined with the original similarity between the document and the query in a weighted sum. Use a development collection to tune the weights.

5. Use the anchor text, where the *idf* of anchor terms is computed as *inverse anchor frequency* i.e. *iaf*. Integrate the similarity between the anchor text and the query in the overall ranking score using the same strategy described for titles above.

6. Augment the document-query similarity score with the pagerank of the document, in a weighted sum combination, as for titles above.

7. Implement postings file compression using one of the two methods discussed in IIR Chapter 5.3: *variable byte codes* or *gamma codes*.

8. Implement session tracking inside a multi-threaded HTTP server.

9. Extract and display a snippet of text for each displayed link. Highlight query terms in the text snippet.

Bonus points will be awarded for any extension implemented beyond the one that is required. The number of bonus points will depend on the complexity of the extension.

Evaluate your implementation on the 10 queries from HW03, plus 10 new queries. Compute Precision@10 and compare with Google's performance on the same queries – use the site search capability through the operator `site:ohio.edu`.

Option 2: OU Web Search Engine, Using Apache Tools

Implement and evaluate a full-fledged search engine for the `ohio.edu` domain using the open source IR tools from Apache: `Lucene Core`, `Nutch`, and/or `Solr`. The implemented engine should index at least 100K pages.

Option 3: Dimensionality Reduction for IR

Implement and evaluate (i.e. try to replicate results) the Rare Term Vector Replacement (RTVR) method, as described in the 2011 paper "*Dimensionality Reduction for Information Retrieval using Vector Replacement of Rare Terms*" by Tobias Berka and Marian Vajtersic. The paper is linked from the class website.

Software Package and Project Report

Turn in a hard copy of your project report on the day of the deadline, in Stocker 337 or in my mailbox. Electronically submit a directory that has your working code and a README file. Create a gzipped, tar ball archive of your directory, and upload it on Blackboard. Your project will be tested on `california.cs.ohio.edu`, consequently all the project data – e.g., crawled files, index – need to reside on that computer. Do not upload project data on Blackboard! The README file should describe in detail how to build and use your search engine, and also the location of the corresponding data files.

For example, if the name is John Williams, creating the archive can be done using the following commands:

> tar cvf williams_john.tar williams_john

> gzip williams_john.tar

These two steps will create the file 'williams_john.tar.gz' that you can upload on Blackboard.

Please observe the following when handing in your project:

1. Structure, indent, and format your code well.

2. Use adequate comments, both block and in-line to document your code.

3. Type and nicely format the project report, including discussion points, tables, graphs etc. so that it is presentable and easy to read. Include a description of all the design decisions that you made that may have an influence on the results. The project report should contain a detailed description of all the modules in your IR engine implementation.

4. Working code and/or correct answers is only one part of the project. The project report, including discussion of the specific issues which the project asks about, is also a very important part of the projet. Take the time and space to make an adequate and clear project report.