Final project, Info 427
Fall 2013
Due: Sunday December 15, 11:59PM

For the final project, choose one of the following two options:

- **the programming option** in which you’ll implement and evaluate your own search engine, using your progress on the assignments so far as a starting point (see Section 1 below), or

- **the research paper option** in which you’ll explore a topic related to web search and write about it in depth (see Section 2 below).

You may work alone or with a partner for the programming option. For the paper option, you must work individually. For either option, please start early, and contact the course staff if you have any questions or concerns about your project.

1 Programming option

If you choose the programming option, your task is to build a fully-functional search engine in Perl. The code you developed for assignments 1 through 4 will be extremely helpful here. The main work on the project will consist of (a) making your code from assignments 1 through 4 work correctly (in case you didn’t get them working the first time), (b) combining the various components together, (c) adding a web interface, (d) conducting some evaluation experiments, and (e) producing a brief report with documentation and experimental results. Compared to commercial search engines, yours will be relatively simple and quite small-scale, but it will include all of the essential elements.

1.1 Search engine specifications

**Crawler.** Use the crawler you developed in the first assignment to crawl a small portion of the Web (say, a few thousand pages). Since this collection of pages will be quite small compared to the trillions of pages on the web, it’s probably a good idea to focus the crawler on one particular topic of interest. To do this, use a dozen or more seed URLs for pages about some topic of your interest, and make sure your crawler is conducting a breadth-first search. For example, if you start from Justin Bieber-related websites, you can name your search engine something like Bieberoogle.

**Indexer.** Use the indexer you developed in the second assignment to parse your crawled pages, filter stop words, stem, extract links, and build the inverted index and various other data structures needed for text analysis.

**Link analysis.** Implement the PageRank algorithm to compute the PageRank of each page in your index, and store this in a data file.

**Web interface.** Develop a Web site that will be the front-end to your search engine. (Check out Perl’s CGI module to help with the web programming. The scripts will run in your CGI directory on Capricorn.) At a minimum, the interface should have a logo, a text field for the query, and a search button. When the user submits a query, he or she should be presented with a ranked list of hits. Each hit should display the title of the page, its URL, and a link to the page itself. This
information is available from the inverted index and other data files created by your crawler and indexer. If there are too many hits (say more than 100), display only the first 100.

**Retrieval and ranking.** When the user submits a query, your search engine should build a ranked list of relevant hits. In particular, it should:

a. Stem the query terms. (You may or may not also want to stop the terms.)

b. Use the inverted index to find the hit list – the set of all pages matching the query. The most basic way of doing this is to find all documents containing all of the search terms, but you might try other options as well (e.g. all documents containing any of the terms, or all documents containing most of the terms, etc).

c. For each page in the hit list, compute the similarity between the page and query using TFIDF.

d. Combine the TFIDF similarity score with the PageRank of each hit, to give a final score to each page. (How you do this combination is again a design decision; there’s no clear correct answer, so we suggest doing some experimentation to find a method that works. You could multiply the two scores, or add them, or take a weighted sum or weighted product, etc. Make sure to explain your choice and how you arrived at it in your documentation!)

e. Finally, sort the hits by the resulting score and present the ranked list back to the user as a dynamic (CGI) Web page.

**Evaluation experiments.** Ask at least two people not involved in your project (i.e. people in another I427 group, or not in the class) to test your search engine and evaluate how well it performs. Exactly how you do this is up to you. For example, you could conduct a quantitative evaluation, in which you measure how long it takes the user to find a page of interest, or what percentage of the time they found a page of interest in the top 10 hits. Then have the user try the same queries on Google and/or a modified version of your search engine that just returns a list of random pages. We would expect these performance measures to be better on your search engine than on the random version, but perhaps not as good as on Google. Alternatively, you could conduct a qualitative evaluation, in which you have the users test the search engine for a while and then answer a survey about their experience (e.g. How often did the search engine return a page of interest? Were higher-ranked documents of more interest to you than lower-ranked documents? etc).

**Further enhancements.** We’ll expect you to implement a bit more than the above minimum requirements – this is an opportunity to be creative! For example, you might implement some additional features typical of modern search engines, like allowing queries for exact phrases, or supporting access to page caches, or even the ability to search for images (like Google Images). You could implement improvements in the retrieval and ranking, by for example using the content of page titles (perhaps weighting them more heavily than terms appearing in the web pages themselves). You might enhance the user interface, by for example displaying just 10 hits per screen if there are more than 10 hits, and giving appropriate navigational links on each page so that the user can move forward and backward through the pages. You could conduct more detailed evaluation experiments, including a comparison to another search engine like Google or Yahoo. Or try something else – maybe you’ll create a new search engine feature that no one has even thought of before!
1.2 What to turn in

Turn in two files, via OnCourse:

1. Your Perl source code. You’ll probably have multiple source files, so combine them into a zip archive file (the `zip` Unix command can do this for you). Make sure your code is legible, understandable, and commented. Please use meaningful variable and function names. Cryptic or uncommented code is not acceptable.

2. A separate text file or PDF with documentation on your search engine. Make sure you cover the following key points:

   (a) Explain how to run the various back-end portions of your engine (the crawler, indexer, and PageRank computation). Please give sample command lines so that we can easily figure out how to run your code.

   (b) The URL of your search engine’s web interface. We’ll be trying it out!

   (c) A description of any design decisions you made throughout the project. For example, how did you find the hit list? How did you combine the PageRank and TFIDF scores?

   (d) The results of your user evaluation experiment.

   (e) Give credit to any source of assistance (students with whom you discussed your assignments, instructors, books, online sources, etc.).

1.3 Grading

We’ll grade based on the correctness, style, and documentation of the code. In particular: Does the code work as expected? Does it use reasonable algorithms and data structures as discussed in class? Did it produce reasonable results? Was it tested thoroughly? Does it check for appropriate input and fail gracefully? Is the code legible and understandable? Does it use subroutines for clarity, reuse and encapsulation? Is the code thoroughly and clearly commented? Is there an adequate readme file?

Extra credit. We’ll give extra credit to assignments that implement more sophisticated indexing features than the ones described here. For example, you could add a feature that could support searching for phrases, either by storing word positions in documents or by indexing n-grams. Please describe in detail any extra credit work in the documentation file (to make sure we don’t miss it while grading).

Academic integrity. You and your partner may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about Perl syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials in the documentation of your source code. However, the code that you (and your partner, if working in a group) submit must be your own work, which you personally designed and wrote. You may not share written code with any other students except your own partner, nor may you possess code written by another student who is not your partner, either in whole or in part, regardless of format. In accordance with University policy, consequences of academic integrity violations may include failure on this assignment or in the course.
2 Research paper option

We’ve covered many topics related to web search in I427, but we’ve really only scratched the surface of what is a large and highly dynamic field. The paper option is an opportunity to explore some aspect of search engines or the search industry in depth, and to write a research paper about that topic. If you choose this option, you must work individually (not in a partnership).

Your paper should be long enough to discuss the topic in depth. There are no particular length requirements, but we would suggest about 3000-5000 words (10-20 pages). Make sure to use a proper academic writing style. References should be listed in a consistent format, and citations to references should be given throughout the paper to back up facts and claims. You may also include original research material (e.g. results of experiments on Google searches), but this is not required.

You may write on any topic of your choice related to search, but here are a few sample topics that could lead to interesting papers:

- Google and other search engine providers maintain huge repositories of information both about the world (e.g. web page caches, scanned books, videos, satellite maps) and about people’s behavior (e.g. search query logs). This has frequently made them the subject of requests for both censorship of their content and for sharing private information about their users, particularly by governments. For example, Google has repeatedly sparred with the Chinese government about whether or not to censor search results that the government considers offensive. What other pressures have search operators faced? How have different companies responded to these requests? What are the legal, moral, and ethical implications of censorship and/or sharing private data about people with third parties?

- Fighting search engine spam is a constant struggle for modern search engines. Spammers create webpages that attempt to artificially raise the ranking of pages by, for example, including lots of irrelevant keywords on a page, or creating hundreds of fake pages whose only purpose is to link to another page and increase its ranking. This has created a sort of “cold war” between spammers and search engines, with one side deploying some new technology, only to have the other side create a new attack or a new counter-measure quickly thereafter. What kinds of techniques do spammers use to game search rankings, and what kinds of techniques have search engines deployed in response? Why are search engines vulnerable to this problem? What might be better, long-term solutions to preventing malicious users from impacting search results?

- The majority of Google’s income comes from selling advertisements that appear on its search engine result pages. How does this ad market work? How does Google decide which ads to show in response to a query? How do advertisers maximize their visibility while paying as little as possible?

- The short (15 year) history of the web search has been tumultuous, with many once-prominent players unexpectedly falling into obscurity (e.g. Lycos, AltaVista), while tiny start-ups have grown into multi-billion dollar companies (e.g. Google, Yahoo). Write a paper that chronicles the history of the web search industry. How has Google emerged as a resounding success, where so many others have failed?
Modern search engines use extremely complex system architectures to be able to analyze, store, and retrieve trillions of web pages in a fraction of a second. Web search applications have driven the creation of new technologies to handle this challenge, including Google's BigTable and Google File System, and Yahoo’s Hadoop. Write a paper describing some of these technologies, their major architectural features, how they developed, and how they are used in modern search engines and other applications.

Another topic of your choice!

2.1 What to turn in

Turn in a PDF of your paper, via OnCourse. Make sure that your paper is written in a proper academic style and includes thorough references and citations.

2.2 Grading

We'll grade based on the breadth, depth, and insight of your research, as well as the quality of writing in your paper. Feel free to consult the course staff for advice about paper topics, or for feedback on drafts.

Academic integrity. We have a zero-tolerance policy for plagiarism. All material you submit must be your own work which you personally and independently completed. If any portion of your submitted material is not your own work, you must make this explicitly clear using proper citations. For instance, all sentences in your paper must be your own unless you place the relevant sentences in quotation marks and give a citation to the source. It is not acceptable to copy or paraphrase even a small part of another work. In accordance with University policy, consequences of academic integrity violations may include failure on this assignment or in the course.