Computing and Information Science
Cornell University

Info 4300 / CS 4300
Information Retrieval
Fall 2012

Assignment 2: Latent Semantic Indexing

Home
Syllabus
Books and Readings
Assignments
Test Data
Examinations
Academic Integrity

About this site

Assignment 2
Due: Saturday, October 6, at 12 noon
Late submission permitted until Sunday, October 8, at 11:59 p.m.

Assignment

The purpose of this assignment is to demonstrate your understanding of latent semantic indexing. For this assignment it is recommended that you write your program(s) in Python but Java, but C++ are permitted.

For the matrix calculations:

- If you write in Python, you can use NumPy, http://docs.scipy.org/doc/numpy/reference/routines.matlib.html, in conjunction with SciPy, http://docs.scipy.org/doc/scipy/reference/sparse.linalg.html#scipy.sparse.linalg.svd. NumPy supplies the matrix foundations, and SciPy has the singular value decomposition.
- If you write in Java, you can use the JAMA package http://math.nist.gov/javanumerics/jama/, which provides singular value decomposition.
- If you write in C++ you will have to find a matrix package that will do singular value decomposition.

For the assignment, you will use the set of files that were used for Assignment 1. They are described on the web page, testData.htm.

A search engine that indexes full text using latent semantic indexing

Write a program *LSI* that does the following:

Building the index(es)

- Create the term-document matrix (as in assignment 1, omit stopwords and use tf.idf weighting, (1 + log(tf)) and log(N/df)). Apply singular value decomposition to create

an LSI representation of the matrix.
- Store the representation of the documents in the concept space.

Searching the indexes

After the indexes have been built, the user interface accepts queries from users. A query is a simple list of terms.

The program should do the following:

- Transform the query vector to the LSI space.
- Use a suitable similarity measure to compute and rank relevant documents.
- For each of the most relevant documents (perhaps three to five), the program should give the score and print out a portion of the document from testData.

Run **three tests** of your program that demonstrate its features. For each test, state which features it is demonstrating (e.g., a test run might show how the program handles a query consisting entirely of stop words).

Varying the number of singular values

The number of singular values chosen (*k*) has a great impact on the quality of retrieval. Select at least three different values for *k* and submit three **test runs** that contrast the results with suitable test data.
(You might wish to check whether the query 'graphene' retrieves document 33, in which it doesn't appear, similarly whether 'kids' retrieves document 04, and whether 'engineering' retrieves document 33.)

Report

Provide a brief report (about one or two pages). Please include your netid along with your name at the top of the report. The reprot should describe: the mathematical basis for your programs, how your program takes advantage of the capabilities of the mathematics package that you used, and which features you have implemented yourself.

You should explicitly describe the effects of varying *k*, the number of singular values chosen to represent the concepts in the set of documents.

The graders will run your programs, but **they will not go into the source code and modify it**. All files and any other options must be specified as input. The report should include a short set of instructions that tells the grader how to run your program, including how to specify the stop list and test data file.

Submission

You should submit a zip archive, named **a2.zip**, that contains the following:

- Program *LSI*
- Output from your test runs
- Report

You do not need to submit the mathematical libraries that you use, such as JAMA, but your report should state very clearly what the graders should do to build and run your programs.

William Arms
Last changed: September 2012