

OER Recommender: A recommendation System for Open Educational Resources and the National Science Digital Library

A report funded by the Andrew W. Mellon Foundation

by Joel Duffin, Brandon Muramatsu and Shelley Henson Johnson, Center for Open and Sustainable Learning

Introduction

Context is critical in teaching and learning of science, technology, engineering and mathematics. *Context* provides the glue that binds individual concepts and activities together into a coherent whole; it may be provided by faculty through a lecture, an assignment or an activity, or it may be provided by a digital learning resource through the structure it provides for activities. *Context* is critical to the use of any resource, especially the National Science Digital Library and the educational digital libraries supported by the NSDL program. Without *context*, the NSDL is little more than a collection of interesting materials.

Context is needed to transform these resources into teaching and learning materials. Higher education in the United States in science, technology, engineering and mathematics (STEM) is characterized by a course structure. A course is typically composed of individual classes, in which activities such as lecturing, discussions, quizzes and laboratory exercises typically occur. While much research has been done recently on improving student learning and improving teaching techniques, the basic course structure has remained relatively constant. Courses, on the whole, are relatively stable; change is measured in terms of years for most courses (e.g., introductory biology, calculus and differential equations, mechanics, electronic signals and systems, etc.). These factors are strengths; the concept of a course provides an understandable structure that is readily apparent to faculty and students in higher education. The course provides *context* for teaching and learning STEM in higher education. OpenCourseWare repositories are catalogs of courses and the course materials that support those courses.

OpenCourseWare repositories represent a large, untapped source of digital learning resources for the NSDL. For example, MIT OCW, as the original OpenCourseWare repository, has approximately 830 courses (65% of the total) in STEM disciplines, and the newly launched USU OCW and JHSPH OCW have five courses respectively in STEM disciplines [23, 48, 14]. As additional institutions develop OpenCourseWare repositories, and as each repository publishes new courses, this source of high-quality, contextualized course materials in STEM has the potential to be an ongoing source of important content for the NSDL. The development of new OpenCourseWare repositories is facilitated with the eduCommons infrastructure software. As MIT and adopters continue to advocate eduCommons' benefits (including its open source nature, scalability and ease of use), it is reasonable to expect that many, if not all, of new OpenCourseWare repositories will build upon Utah State's eduCommons infrastructure. The adoption of this infrastructure, coupled with the proposed work, provides a straightforward pathway to opencourseware materials for the NSDL and will greatly facilitate the ability of the NSDL to tap this growing resource.

The educational digital libraries of the NSDL and OpenCourseWare repositories have complementary strengths in the content they typically contain. The educational digital libraries of the NSDL contain a wide range of materials, but are typically discipline-based collections of “stand-alone” resources such as applets, simulations and interactive software (e.g., ComPadre’s collections of physics digital learning resources, NEEDS—A Digital Library for Engineering Education, BioSciEdNet, etc.). The courses in OpenCourseWare repositories provide a traditional structure (lectures, labs, discussion sections, assignments) for teaching and learning STEM in higher education.

The focus in educational digital libraries has typically been on providing access to tools to supplement or in some cases replace traditional activities in teaching and learning in higher education. These resources have typically been considered “separate” from traditional classroom activities and are cataloged as independent resources. Educational digital libraries have developed a number of value-added services to help provide additional *context* for the digital learning resources in their collections.

OER Recommender

OER Recommender (<http://oerrecommender.org>) is a service that helps people find relevant open education resources. When a person browses a web page in the NSDL or in registered OpenCourseWare repositories, the recommender annotates the page with “related resources” links. Plans are to add functionality to the recommender so that it will give user specific recommendations via email and the recommender website. User specific recommendations will be based on resources that a person previously visited, shared, rated, tagged, and otherwise paid attention to.

Exploration

The core technology in the recommender is document clustering. Document clustering is an automated process for finding related documents by calculating the distance between the documents in a collection. When designing the recommender, a [latent semantic analysis](#) (LSA) approach to clustering was initially considered because it offers the benefit of not basing the calculation of the distance between two documents solely on the overlap of words that they share. For more information about LSA, see [Handbook of Latent Semantic Analysis](#), [LSA online resources](#), and the [lsa module for R](#). The recommender does not use LSA because it was considered be too complex to implement and too computationally expensive to use. In addition, the standard [Term Vector Model](#) approach that the recommender uses can be built upon to implement an LSA approach.

Implementation

In order to provide recommendations, the recommender uses metadata gathered by the feed harvester written for [ozmozi](#). The recommender was able to reuse the ozmozi harvester to gather atom metadata exposed by the OCW repositories indexed by [OCWFinder](#). In order to gather metadata from the NSDL, the harvester was extended to support the [OAI](#) protocol. In order to provide recommendations, the recommender accesses aggregator databases to identify related resources. The recommender calculates a similarity score for pairs of resources based on the words in their titles, descriptions, and tags. For each resource, the recommender stores in a *related resources* table, the resources with the highest similarity scores. The recommender uses four phases calculate

recommendations: parse metadata, calculate local term weights, calculate global term weights, and calculate similarity scores.

Parse Metadata

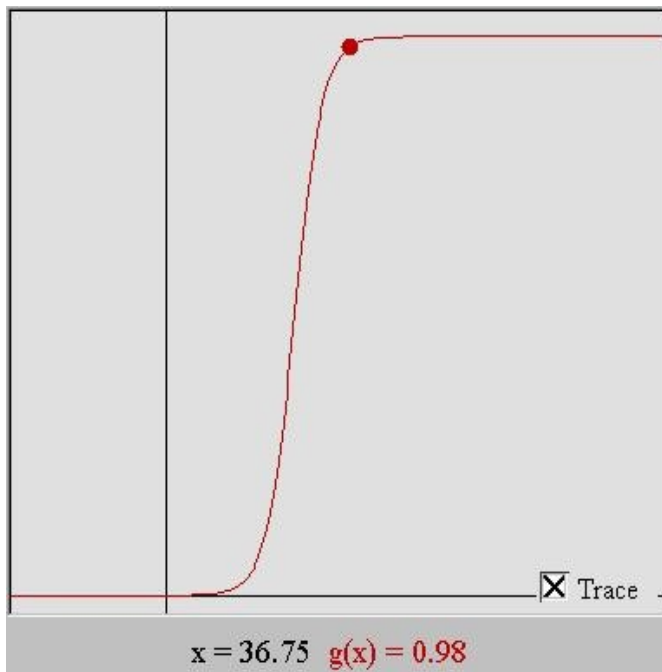
Using a string tokenizer the recommender breaks metadata text into terms. It throws away stop words (common words such as ‘and’, ‘is’, ‘of’ that do not add meaning). Next the recommender converts terms into their stems using the [Porter algorithm](#); for example ‘running’, ‘ran’, and ‘runs’ all get collapsed into ‘run’.

Calculate Local Term Weights

A local term weight is a measure of how important a word is for describing a document. The more frequently a word appears in a document, the more important it is... up to a point. Of course, where a word appears in a document is probably a good indicator of how important the word is as well. For example if a word appears in a title, it is probably more important than if it appears in the body. One more important factor to consider is document length (total number of terms in the document); all other things being equal, the longer a document is, the more times a term will appear in it. So the recommender normalizes term frequencies using the document length.

To calculate local term weights the recommender uses a function that looks like the one shown in Figure 1.

Figure 1. Local Term Weights Function.



The x-axis represents term frequency and the y-axis represents the local term weight. The more time a term appears in a document, the more weight it is given. However after reaching a threshold, it plateaus. This is especially important in situations where

document creators might be trying to game the system. The formula used in the recommender is:

$$ltw_i = 1 / (1 + (e^{(.0044)dlen}) \cdot .7^{(ft_i - 1)})$$

ltw_i is the local term weight for a given term in a document.

$dlen$ is the total number of terms in the document.

ft_i is the number of times a given term appears in a document.

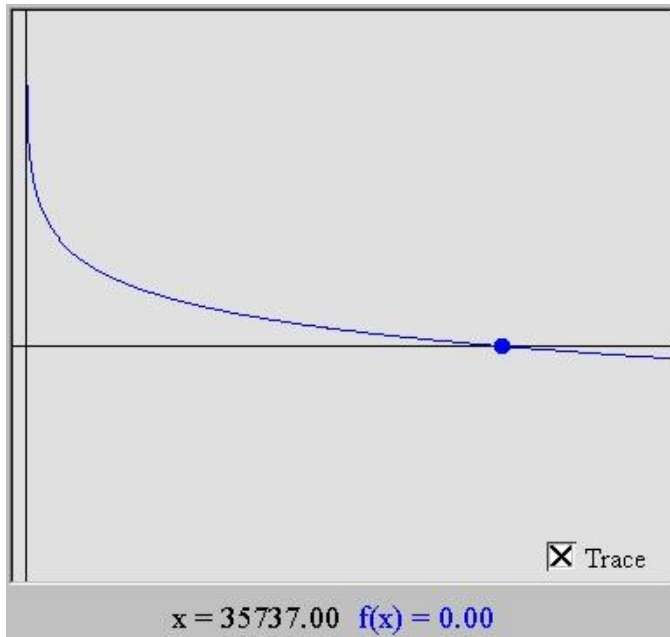
Note that the constants .0044 and .7 in the equation determine the shape of the curve. Those values were chosen based on their use in the [MeSH system](#). As explained there, the values should be tuned to the target data set using an empirical approach. We played with the parameters some and the values they used seemed fine, so we adopted them.

Calculate Global Term Weights

Global term weights are a measure of how important a word is for distinguishing documents within a collection. The more documents a word appears in, the less value it has for characterizing documents. For example, the term 'USU' appears in every metadata record for resources in USU's OpenCourseWare. As a result, it has no value for characterizing clustering resources. The term 'USU' becomes in a sense, a stop word, similar to those thrown away during the parse phase. To calculate the global term weight for a term, the number of documents that it appears in are counted as well as the total number of documents.

To calculate global term weights OER Recommender uses a function that looks like the one graphed in Figure 2.

Figure 2. Global Term Weights Function.



The x-axis represents the number of documents that a term appears in. The y-axis represents the global weight assigned to the term. The formula used is:

$$gtw_i = \log (D/df_i)$$

gtw_i is the global term weight for term.

D is the total number of documents.

df_i is the number of documents that the term appears in.

See [Mi Islita's explanation of Term Vector Theory](#) for details.

Calculate Similarity Scores

Once local term weights and global term weights are calculated, recommendations can be created. To create recommendations for a document the recommender first finds all documents that have any of the same terms in it. This can be a very large number of documents (e.g. 40,000 in the databases used by the recommender). For each pairing of the document being considered and a document with matching terms we calculate a similarity score. Because calculating 40,000 scores can take a long time (about 15 seconds in our current system), the recommender shortens the list to consider to 200. It does this by sorting the pairs according to the number of overlapping terms. To calculate the similarity score for a pair of documents, the recommender sums over all of the terms that the documents share in common. The contribution from each term is a combination of the local term weight in the first document, the local term weight in the second document and the global term weight. Because in OER recommender, each feed can be considered a separate collection, the recommender calculates global term weights for each of the feeds and uses them to calculate the similarity score.

To calculate the contribution of an individual term to the similarity score, the recommender uses:

$$sst_i = (gtw_{1i})(ltw_{1i})(gtw_{2i})(ltw_{2i})$$

sst_i Is the contribution to the similarity score for a given term.

gtw_{1i} is the global term weight from the feed that the first document is in.

ltw_{1i} is the local term weight from the first document.

gtw_{2i} is the global term weight from the feed that the second document is in.

ltw_{2i} is the local term weight from the second document.

See [NCBI's explanation of how MeSH calculates Related Articles](#) for details. Once the recommender has calculated similarity scores for the 200 documents, it sorts the documents by those scores and stores the top 10 in a database.

Displaying Recommendations

Once recommendations are stored in a database, displaying them to the user is straightforward. The recommender provides a [Greasmonkey script](#) that requests recommendations for web pages that a user browses. If any related documents are returned, the script inserts HTML for the recommendations into the web page. The eduCommons team has built a plone tool so that anyone running eduCommons can turn on recommendations. That way users do not have to have install the Greasemonkey script in order to see recommendations on the eduCommons website. The OER Recommender web site also provides a Javascript include that others can use to easily add recommendations into their websites.

Getting Resources into the Recommender

Currently OER Recommender only provides recommendations for URLs that it has metadata for, so in order to include resources in the recommender, a metadata feed must be registered. You may register an OCW site that provides an RSS feed by going to the [OCW Finder](#) website. If you have an OAI feed or an RSS feed for learning objects or other OERs, you can add your metadata by sending a feed title, URL, and display URL (home page of the repository) to oerrecommender AT cosl DOT usu DOT edu.

Future Directions

There are many things left to be done: (1) adapt recommendations by monitoring which recommended resources people click on and how long they stay at recommended resources, (2) provide links to more recommendations (so users can see more than the default 5), (3) provide a way for people to indicate when a recommendation doesn't work, (4) provide a way for people to register and login, so they can receive personalized recommendations (perhaps via email), (5) create a process whereby recommendations can be updated without going through all the documents in the database, (6) add functionality for retrieving recommendations for arbitrary web pages (recommending on

demand) perhaps via a bookmarklet, (7) create whatever Greasmonkey-like add on is supported on IE. There are also plans to integrate the recommender with the aggregator and feed reader to provide recommended news items based on the attention people have previously paid to news articles.