

# An Automated Management Tool for Unstructured Data

Maciej Ceglowski  
National Institute for  
Technology and Liberal  
Education  
mceglows@middlebury.edu

Aaron Coburn  
National Institute for  
Technology and Liberal  
Education  
acoburn@middlebury.edu

John L. Cuadrado  
National Institute for  
Technology and Liberal  
Education  
drjlc@acm.org

## Abstract

*The rapidly growing quantity of online data has created a need for automated, content-based categorization and search tools. The authors describe an open-source, Web-based archive management which uses latent semantic indexing, coupled with vector clustering techniques, to provide users with a fully searchable and automatically categorized interface to a data collection. The default English document parser included in the project uses part-of-speech tagging and recursive maximal noun phrase extraction to create a more effective term list for LSI than traditional stop list techniques. The archive interface supports multiple user views of the data collection. Advanced search features are implemented through relevance feedback, and do not require users to learn a query syntax.*

## 1. Introduction

The National Institute for Technology and Liberal Education is an organization devoted to making information technology usable by educators and students who may not have a technical background. Through the generous sponsorship of the Andrew W. Mellon foundation, we have recently been able to pursue research in Web-based tools for managing large collections of unstructured data.

Researchers in the sciences and humanities are faced with a growing influx of source material in electronic form, much of it with little or no associated metadata. While these users have expert knowledge within their own field, and require sophisticated information retrieval capabilities to make effective use of their data collections, they may have only basic familiarity with search engines and computer interfaces. Our challenge has been to apply information retrieval and categorization algorithms in a way that is transparent to end users and corresponds to their own perceptions of relevance in the target domain.

Over the past year, we have worked to create a scalable, cross-platform web application suitable for use in organizing complex collections, and usable with

minimal training. We assume that the raw documents available to the system are homogenous with regard to language, and lack any metadata whatsoever. The task at hand is impose as much structure on this collection as possible through automated means, both by making the collection searchable, and by organizing it into topical categories acceptable to end users.

The tool described in this paper uses a combination of latent semantic indexing (LSI) and vector-space clustering techniques to achieve these goals.

## 2. Building a Collection

We assume that any documents added to the repository are in plain text or in a tagged markup (HTML or XML) format. Documents are scrubbed using a suite of Perl regular expressions to find paragraph boundaries, strip markup, and remove footnotes and page headers.

Traditionally, LSI has relied on a combination of stop lists and word stemming to generate lists of semantically useful words from each document in a collection [1]. Words not excluded by the stop list are typically stemmed by a language-dependent algorithm that reduces the number of variant forms.

The stop list approach has two weaknesses: first, it does not account for homonyms and polysemes, especially words that exist both as verbs and as nouns, a very common feature of English (e.g. ‘fire’, ‘fly’). Second, it fails to identify semantically meaningful multi-word constructs, such as proper names and noun phrases (‘president of the cost committee’). This is especially problematic for collocations, such as ‘hot rod’, that differ in meaning from their constituent words.

## 3. Part of Speech Tagging

Our own approach for English texts is to use a part-of-speech (POS) tagger in tandem with a regular expression for finding maximal noun phrases in tagged text [2]. This allows us to find all nouns and noun phrases in the text, while discarding other parts of speech presumed to carry less semantic content.

Our POS tagger [3] uses a stored lexicon of 44,000 English words. This lexicon contains statistical information on part-of-speech usage drawn from the Penn Treebank Project [4], an annotated corpus of text developed by the Linguistic Data Consortium. The tagger uses stored knowledge of the relative probability that a given word will fall in a given grammatical category, along with a bigram Hidden Markov Model of POS occurrence. The tagger assigns a tag  $t_i$  to a word  $w_i$  according to the the following formula:

$$t_i = \underset{j}{\operatorname{argmax}} P(t_j | t_{i-1}) P(w_i | t_j)$$

If the tagger does not recognize a word, the HMM is supplemented by a metric based on word morphology.

By recursively applying the maximal noun phrase regular expression derived from [2] to each tagged sentence, we are able to add all valid multi-word noun phrases in the document to our term list.

#### 4. Latent Semantic Indexing

We create our vector space data model by generating a standard term-document matrix (TDM), with the caveat that a term in our model can consist of a multiple-word proper name or noun phrase.

Values in the TDM are weighted according to the following formula:

$$x_{ij} = g_j \ln(f_{ij}) n_i c_j$$

Where  $x_{ij}$  is weighted value for term  $j$  in document  $i$ ,  $c_j$  is the number of words in the term,  $f_{ij}$  is the occurrence count for term  $j$  in document  $i$ ,  $g_j$  is a global term weight, and  $n_i$  is a normalization factor. All document vectors are normalized to unit length to compensate for variations in document size.

The collection is indexed using the general LSI technique described in [1], retaining 100 singular values from the SVD computation. This intentionally conservative value is chosen to reduce the occurrence of false positives in our search result set, as users have shown a low tolerance for these kinds of errors.

#### 5. Search Services

Our search interface consists of a web page with a single text input field. Users are not expected to have any knowledge of Boolean search syntax for conducting advanced searches. Instead, the system relies on various forms of relevance feedback.

The most basic type of feedback is a 'find similar' link associated with every document in a result set. Users can click on this link to run a similarity search against any document, and receive a ranked result list of the documents that are semantically. This search,

implemented as a simple comparison against the stored, scaled document vector, allows for rapid horizontal navigation across related results.

A second level of relevance feedback is provided by a 'document basket', a temporary session-based document store. Users can add or remove stored documents at any point, as well as run a similarity search against the aggregate of all stored documents. This feature allows users to refine a query by giving the search engine examples of relevant documents.

An additional level of relevance feedback takes the form of a list of seven related terms displayed at the head of every result set. This list is generated analogously to the list of returned documents, through a comparison with stored term vectors, and represents the terms semantically closest to the query. These nearest terms are clickable, for maximum ease of use. This display of related terms helps orient users in the search engine's semantic space, and gives them a general feel for the scope and focus of the document collection.

A final, subtle form of relevance feedback is the user's ability to enter natural language queries of arbitrary length. In practice, this enables users to paste in snippets of text from stored documents, Web search results, or local files, along with running more traditional single-word queries.

The current search interface has proven valuable in trials with users with small, coherent text collections. Even in cases where the user is familiar with every document in the collection, the ability to quickly search and navigate by means of 'find similar' links enhances the perceived value of the collection.

#### 6. Automated Categorization

As document collections grow in size, it becomes important to impose additional structure on the collection to keep it navigable. The goal of our auto-categorization efforts is to partition large document collections into categories meaningful to a human user based on their content, and without requiring human intervention. We then use this clustering as a starting point for domain experts, providing them with an interface that allows them to edit and extend the machine-generated clustering based on their own judgment and knowledge.

For text collections, this ability to auto-categorize based on content is a convenience, since it spares the user many hours of reading and metadata creation. However, when applied to non-text collections, the clustering becomes a source of testable hypotheses about document similarity which a human examiner would not necessarily be able to infer from an examination of raw document data [5]. This type of pattern discovery can be of particular use in fields like protein structure analysis, where only a small subset of documents may have been accurately characterized by laboratory experiments.

The TDM generated earlier for latent semantic indexing also serves as the starting point for clustering our collection. Automated clustering requires both a similarity measure, and a metric for evaluating cluster quality. The distance measure is a configurable part of the system; by default, we use the inner product between document vectors. The program runs the clustering algorithm over a range of centers, and evaluates each clustering with the silhouette metric [6] to determine an optimal (or acceptable) number of centers to use in the actual partition of the collection.

For very large or incoherent collections of documents, the clustering may produce clusters of unequal size. It is possible to recursively cluster the collection until all clusters fall below a set size threshold.

Empirical tests on real world document sets give encouraging results. In a recently-created search interface for Civil War articles digitized by the University of Virginia [7], automated clustering was able to find boundaries between topics like secession, the debate over the Northwest Territories, and accounts of the Battles of Gettysburg and Antietam. Similarly effective results have been observed on a collection of author's notes developed for the popular science writer Steven Johnson (author of *Emergence* and *Interface Culture*), who helped test an early version of the system in his work on a forthcoming book on the brain [8].

## 7. User Interface

As automated clustering is not expected to be error-free or comprehensive, extensive provisions are included for allowing human users to interact with, delete, name and restructure the computer-generated categories, as well as create categories of their own

Users are presented with a three-column interface containing a list of clusters, a list of documents in the working cluster, and a view of the document itself [9]. Computer-generated clusters are given provisional names derived from term frequencies in an attempt to give the user an idea of the cluster content. These names can be edited by the user, who can also assign more verbose descriptions to each cluster. Users navigate through clusters by clicking on document and cluster titles. The interface provides visual cues about the relative size of clusters, and allows users to examine documents within them.

This interface is tightly coupled with the search features described above, in an effort to make cluster management as efficient as possible. All documents have a 'find similar' link displayed with them, and a text search box is always visible. Users have the ability to create new clusters from a search result set, or from the contents of the 'saved documents' basket, with one click. Every cluster displays a 'find similar' search link.

## 8. User-defined Views

In real world collections, there will never be a single correct clustering of the data. Expert users may apply out-of-band knowledge to the clustering, or they may organize it differently based on their research interests. The topical clustering described for the Civil War collection may be useful to the lay reader, but not to a historian looking to organize the articles by region, or a student of journalism interested in variations in writing style.

We allow for multiple user views of a document collection. Information on cluster membership is stored for each user, and kept separate from the document data itself. The stored views are persistent, allowing users to customize them over time, and add their own domain knowledge to the system. These custom views become a source of additional information about the collection, since they can be compared and analyzed in relation to one another as well as in relation to the original computer-generated clustering.

## 9. Architecture

The archive management tool is divided into two separate components, which communicate with one another via XML-RPC. Each component is implemented in Perl, with computationally intensive sections (such as the singular value decomposition algorithm) included as compiled C extensions to the Perl core.

The first component serves both as the main document repository, and as a search server for XML-RPC enabled search engine front ends to the collection. This component also stores information about the canonical, computer-generated clustering that serves as a template for newly-generated user views

The second component of the system is a database-driven Perl application that stores and manages user views. This component handles authentication, logging, and stores information on individual users and their views of the collection. The component does not store any document data, which is always requested from the master server.

The user interface to both components is implemented as a Web application, loosely coupled to both the archive and search servers. This architecture leaves room for a possible future desktop front-end to the system, with a more sophisticated user interface.

The primary document repository is configurable, allowing for the use of multiple parsers, weighting schemes, and clustering algorithms. This flexibility makes it possible to index of a large variety of text collections in multiple languages, as well as non-linguistic data like biological sequences. Each collection is associated with a parser responsible for generating term lists and parsing incoming text queries. The ability to

index documents in a given language is constrained only by the availability of an appropriate parser.

Both the repository and user view components run on standard PC hardware (i386). Indexing times for collections in the 5-10 K document range are typically under ten minutes.

## 10. Future Work

Several areas of the project are currently undergoing active development:

### 10.1. Graph-Based Search Algorithm

Preliminary work with *spreading activation search* [10], a graph-theoretic analogue to LSI, has shown great promise in providing the expanded recall and relevance feedback features of LSI at a far lower computational cost. Spreading activation search also offers a framework for incorporating hyperlinks and citations into the document model, as well as allowing unlimited changes to a live document collection. Given these advantages, we have shifted our development efforts to the graph-based implementation.

### 10.2. Summarization

One shortcoming of automatic categorization is the difficulty in assigning meaningful category names to automatically generated clusters. Our efforts in topicalization seek to auto-generate a short natural language summary of each cluster based on the content of its constituent documents. A cluster summary would free users from having to skim through every document in a cluster to understand the organizing principle at work, and assist them in evaluating the quality and coherence of the computer-generated cluster.

### 10.3. Visualization

The current search interface relies heavily on text and HTML elements, a constraint imposed by the immaturity of cross-platform web technologies for manipulating graphics. Recent advances in the implementation of the scalable vector graphics (SVG) standard, however, give us hope that we can implement a useful graphical interface to the archive management tool, including desktop-like drag-and-drop capability between clusters, and improved visualization of document similarity through two-dimensional projections. A prototype graphical user interface based on SVG already exists, but cannot yet reliably duplicate all of the features of the text interface.

## 10.4 Biological Sequence Data

Our recent work on protein sequences, outlined in [5], has shown that latent semantic indexing coupled with auto-clustering techniques, has great promise in the area of protein characterization. Structure prediction from sequence is a major open problem in bioinformatics, and vector-space approaches like LSI may offer an improvement over current heuristic and HMM-based predictive techniques.

## References

- [1] Deerwester, S., Dumais, S., Furnas, G., Landauer, T. and Harshman, R., "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, 1990, pp. 391-407.
- [2] Bader, R., Callahan, M., Grim, D., Krause, J. and Pottenger, W., "The role of the HDDI™ collection builder in hierarchical distributed dynamic indexing", *Workshop on Text Mining*, Chicago, 2001, pp. 23-30.
- [3] Lingua::EN::Tagger, a Perl part-of-speech tagger for English text. <http://search.cpan.org/author/MCEGLOWS/>
- [4] Marcus, M., Marcinkiewicz, M. A., Santorini, B., "Building a large annotated corpus of english: the penn treebank", *Computational Linguistics*, 1993, pp. 313-330.
- [5] Ceglowski, M., Cuadrado, J. L., "Using latent semantic analysis in bioinformatics", *O'Reilly Emerging Technology Conference*, 2002. [http://conferences.oreillynet.com/cs/bio2003/view/e\\_sess/3406](http://conferences.oreillynet.com/cs/bio2003/view/e_sess/3406)
- [6] Kaufman, L. and Rousseeuw, R., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York, 1990.
- [7] Valley of the Shadow project search engine. <http://mojave.cet.middlebury.edu/lsi/uva.pl>.
- [8] Steven Johnson Research Notes. Archive and search engine. <http://mojave.cet.middlebury.edu/lsi/sbj/notes.pl>.
- [9] Ceglowski, M., Cuadrado, J. L., Yu, C., "Managing unstructured data with latent semantic indexing", *CNI Spring Task Force Project Briefing*, 2003. <http://www.nitle.org/cni/>
- [10] Preece, Scott. "A spreading activation model for information retrieval", University of Illinois, Urbana, IL 1981.