Fall 2006

A Machine-Aided Approach to Intelligent Index Generation

Shelly Candita Lukon

A Machine-Aided Approach to Intelligent Index Generation:

Using Natural Language Processing and Latent Semantic Analysis to Determine the Contexts and Relationships Among Words in a Corpus

A Thesis

Presented to the Faculty

of the Graduate Program in Computational Mathematics

Department of Mathematics and Computer Science

McAnulty College and Graduate School of Liberal Arts

Duquesne University

in partial fulfillment of

the requirements for the degree of

Master of Science

By

Shelly Candita Lukon

November 27, 2006

Advisor: Dr. Patrick Juola,

Associate Professor of Computer Science

PREFACE

When I was first approached by Dr. Patrick Juola to help build a "killer app" for the digital humanities, specifically a back-of-the-book indexing application based on his paper, "Towards an Automatic Index Generation Tool" (Juola, 2005), I have to admit that I knew very little about what was involved in creating a back-of-the-book index. In fact, it's something that I had just taken for granted as being part of a book, never giving much thought to how it got there, or what criteria determined whether an index was good or poor.

Dr. Juola's ideas for an automated, machine-aided tool for the indexing industry have at their core a mathematical technique known as Latent Semantic Analysis. LSA is a type of factor analysis that we use to determine the context of words in a corpus, and to represent these terms numerically, in such a way that their relationships to other words can be determined, and can consequently be used in structuring the back-of-the-book index. We combined LSA with word sense disambiguation and hierarchical cluster analysis techniques to create what we feel is at least a good first attempt at automating the intelligent portions of the indexing process and generating an index.

We have taken our work to conferences in two different countries ("Digital Humanities 2006" in Paris, and the "Canadian Symposium for Text Analysis (CaSTA)" in New Brunswick), published our preliminary design and results in the proceedings of these conferences, and have also submitted our work to leading journals in the field of humanities computing (including *Literary & Linguistic Computing*). There has been a lot of interest in our application, and we have plans to continue this research in the future. Specifically, we have applied for a grant from the National Endowment from the Humanities to help fund our continued work, and plan to adapt our application to recognize XML-tagged text and other formats, and to extend it to the indexing of other types of media.

ACKNOWLEDGEMENTS

I would first and foremost like to thank my graduate advisor, Dr. Patrick Juola, for providing me with this great opportunity, not only to work on new ideas of publishable quality, but to challenge me beyond what I thought I was capable of. His and Jodi Affuso's support and encouragement through this project have been invaluable.

I would also like to thank my committee members, Dr. Kathleen Taylor and Dr. John Kern, for taking an interest in my project and helping me through the thesis process, and providing valuable feedback about my thesis.

I would like to thank my graduate advisor from the University of Pittsburgh, Dr. Doug Metzler for his input and suggestions at the onset of this project, and for introducing me to Dr. Judith Jablonski, of Pitt's School of Library Sciences, who gave us great assistance and suggestions as we prepared our grant proposal. I am indebted to her and to Ms. Kimberly Henry for providing compelling Letters of Support for our NEH grant proposal.

I am grateful to Dr. Hugo Liu of MIT Media Lab, the creator of the MultiLingua natural language processor that I am using to tag parts of speech in my prototype system. He provided assistance early on in integrating MontyLingua with my application, when I encountered technical glitches.

Many thanks go out to the attendees of the Digital Humanities (ACH/ALLC) and CaSTA conferences for their input and interest in the system.

Of course, I owe the greatest debt to my family, friends, and everyone who has stood by me throughout what has proven to be a long, but extremely rewarding period at Duquesne, in particular during the past year as I worked on my thesis and prototype application, and prepared for the conferences, publications and grant proposal that sprang forth from it.

TABLE OF CONTENTS

Note: Source code will be made available from the following directory on the Duquesne University website: http://www.mathcs.duq.edu/~juola/sclukonthesis/.

LIST OF TABLES

LIST OF ILLUSTRATIONS

INTRODUCTION

Back-of-the-book indexing is the process of generating a list of relevant terms, sub-terms and cross-references from a corpus and providing the user with page references for these terms. It differs from web indexing in its ability to identify synonyms (cross-references) and subcategories (sub-terms). These word relationships enable the reader to understand the overall intellectual structure of the body of text. Back-of-the-book indexing also differs from web indexing in its static nature. A web index dynamically changes in response to a user's query, but the back-of-the-book index imposes a rigid conceptual structure, proactively defining the structure of the response to the user's query. It is in a sense an intellectual précis of the corpus. The index is a fundamental concept in humanities research, and consequently is of great interest to researchers, including this author, in the field of humanities computing.

All indexes allow the reader to easily facilitate direct information retrieval, but a good index also distinguishes between *substantial* (i.e., significant) and *passing* (i.e., one-time) mentions, can provide terminology not explicitly present in the text, and can filter information in such a way as to avoid reader "burnout." In other words, a good index is much more than a simple paper-based search tool.

This indexing process has become somewhat automated through computer applications, most of which can at best generate a concordance, or list of keywords in the text, and serve to reduce the mundane portions of the indexing process, such as organizing the index information that has been gathered manually and providing easy formatting of this information. The difficulty lies in the ability to make intelligent decisions regarding which words or phrases to include, and how to structure them. None of the existing software tools determines which terms to index, nor do they capture context-sensitive information about the terms and their relationships to one another. Human indexers primarily do these steps, which can be extremely arduous and time-consuming. The Chicago Manual of Style estimates that it takes one week to manually index one hundred pages of text (Chicago, 2003). To the author's knowledge, there is no system currently available that can significantly reduce the time required by the human indexer by generating a list of potential terms to index beforehand.

The challenge, then, is to develop a software program that bridges the gap between computerized concordances / editors and manual indexing. This is accomplished by providing a much more robust draft index, which a human indexer can refine in a fraction of the time. The prototype application, an extension of the ideas put forth in Juola (2005), is unique in its ability to incorporate the intelligent portions of the indexing process. Because of this, it can, in fact, meet the challenge.

My application takes advantage of an existing natural language processor to tag terms with the appropriate parts of speech. Once the terms have been tagged, one can easily identify the nouns and noun phrases. This becomes the starting list of terms to index. I determine the frequency of each term, and set upper and lower frequency bounds to determine which terms to include. Next, I construct a matrix whose rows are the terms in the corpus, and whose columns are the "documents" or paragraphs that contain these terms. Applying weightings to the values in the matrix, I quantify the relative importance of each term with respect to each document (paragraph) and to the entire corpus. By comparing each term to the others in the corpus and computing the corresponding covariances, a new covariance matrix is created, which provides the input for the next stage, which uses a multivariate statistical technique known as Singular Value Decomposition to derive the semantic meanings of the terms in the corpus. SVD is based on the principles of Latent Semantic Analysis (Landauer, 1998 and Wiemer-Hastings, 2004) and Latent Semantic Indexing (Deerwester, 1990), which are used to study the relationships among terms and their context. The data in the LSA/LSI literature demonstrates their ability to model human semantic knowledge. They are able to identify words that are similar in meaning without the terms having to be literal matches, and to consistently identify synonyms and antonyms.

SVD is used to perform a decomposition of the covariance matrix into several component matrices. The original matrix can be reconstructed using a subset of these matrices, decreasing the overall dimensionality of the solution, and also reducing noise. The reconstructed data shows us how various terms correlate with one another in the semantic space. The proximity of terms to other terms allows us to make inferences as to which terms appear to be synonyms (correlating

highly in every dimension), and which words seem to be subcategories of broader terms (correlating highly in one or more dimensions.)

Once the term relationships have been identified, the system performs word sense disambiguation to differentiate among word tokens that are spelled identically but have different meanings, and hierarchical cluster analysis to group together words that are semantically similar (and thus, good candidates for synonyms or sub-terms.) The system then generates a structured hierarchical index, sorted alphabetically by main terms, subcategorized where appropriate, listing cross-references where applicable, and enumerates the page numbers corresponding to each of these groupings.

This combination of semantic and cluster analysis is a new and as-yet untried approach to indexing, although it has a history of successful use in other linguistic problems. It is my hope that this approach to creating an all-inclusive, "one-stop shop" for back-of-the-book indexing needs, will revolutionize the indexing industry by providing an easy to use, accurate means of generating an index in a drastically reduced amount of time.

PROBLEM TO BE SOLVED

The challenge existing today around the generation of back-of-the-book indexes is to create a software application that bridges the gap between automated concordances / editors, which perform the mundane portions of the indexing process, and the human indexer, who performs the rather time-consuming intelligent portions of the process.

Therefore, the specific problem that my academic advisor and I identified was the enormous amount of time required to generate an index. By creating a software application that can perform the intellectual portions of the indexing process, thus minimizing this labor-intensive effort, we were able to address what might be argued as one of the greatest challenges in preserving and providing intellectual access to humanities resources.

Before we can construct an application that meets this challenge, we need to identify these intelligent tasks. Juola (2005) identified six basic cognitive tasks involved in constructing a good index, and notes that most of these are currently performed by the human indexer, rather than by any existing software packages. These tasks are:

1. Identification of terms to index (in our case, nouns and noun phrases, eliminating those that are too frequent or too infrequent)
2. Location of all informative references (terms that are contextually significant, as opposed to passing references)
3. Identification and location of synonymous terms (correlate/cluster near each other in a high-dimensional *semantic space*, which is a multidimensional representation of all of the terms in the corpus, where terms that exist near each other in this space are more semantically similar than those that lie farther away)
4. Splitting of index terms into sub-terms (take tokens for each term and disambiguate these into sub-terms)
5. Development of cross-references within the text (mapping terms to other related terms determined to be semantically similar)

6. Compilation of page numbers (and putting the index into a final format that is acceptable to indexers / publishers)

Juola proposes that advances in corpus linguistics, coupled with the development of part-of-speech tagging software, the availability of semantic networks, and the application of factor analysis techniques, can be combined in an multi-stage iterative framework that packages all of the aforementioned cognitive tasks into a single indexing application.

The current state-of-the-art only addresses the sixth of these cognitive tasks. My prototype system automates the rest of these cognitive tasks, using a combination of techniques, many of them taken from corpus linguistics and search engine technology. These techniques include part-of-speech tagging, frequency analysis, Latent Semantic Analysis (LSA), Word Sense Disambiguation (WSD), and Hierarchical Cluster Analysis (HCA). My application aids the human user in making intelligent decisions by providing a robust first draft of the index while allowing the user to intervene and revise / refine the resulting index throughout the process.

In addition to Juola's list of cognitive tasks, there are other resources currently available that define what makes a good index. One such resource is the NISO technical reference, "Guidelines for Indexes and Related Information Retrieval Devices" (Anderson, 1997). Anderson defines an index as "a systematic guide designed to indicate topics or features of documents in order to facilitate retrieval of documents or parts of documents," and states that an index should include the following components: terms representing the topics or features of the corpus, a syntax for combining terms into index headings, creating cross-references among synonymous terms, a procedure for linking headings with documentary units, and a systematic ordering of the index headings. It is the objective of our application to successfully incorporate each of these components.

BACKGROUND

One of the first questions that I asked myself before designing this system was, "Why is it important to create an intelligent machine-aided indexing tool in the first place?" I believe that there is a great need among indexers, academics and professionals of all disciplines to have more advanced tools for providing the best and most efficient access to the information contained in a text. Juola (2006) describes the back-of-the-book index as "an encapsulated and stylized summary of the intellectual structure of the book itself." It does no justice for a great book to have a poor index. The user would be at a great disadvantage in finding all of the relevant information contained therein. I wanted to avoid creating a poor index that is nothing more than a concordance. I am striving with this application to add a level of robustness that is missing from more conventional indexing tools (or at least, missing until a human intervenes.)

Several factors provided the rationale for this project, in identifying the necessity for such an application and for guiding me toward the chosen approach to addressing the back-of-the-book indexing problem.

First, I needed to identify the needs of the human indexers by communicating with them personally. In November 2005, I conducted interviews with several professional indexers (PIs), whose names and email addresses I obtained from the *Politics and International Relations Indexing* group's website (http://www.wave.net/upg/pisig/find%20indexer.htm). I posed the following questions to the PIs:

1. What do you feel are the biggest challenges for an indexer?
2. What (if any) software packages do you currently use to assist you with back-of-the-book indexing?
3. Do you primarily use software that creates copy-ready indexes from your tagged/marked documents (identified with keyword/location pairs)?
4. Do you use concordance-generating software or other searching tools to help you to find the number of occurrences of various words in a body of text?

5. What do you feel are the strengths and the weaknesses of the indexing software packages that you have used?
6. Do you have any questions or comments for me?

Their responses confirmed the excessive time required to create an index, and also vocalized a perceived lack of return on this investment of effort and time. I found that the biggest challenge for most of these PIs was making a respectable living while being paid relatively little in return for performing a very time-consuming task where the time required increases proportionally with both the degree of quality and detail of the index, and with the size of the text being indexed.

Most of the PIs interviewed do not use any single software application to deliver them their final copy-ready end product – they use a variety of methods, including the applications described above, in conjunction with basic Word and PDF searching, and manual cutting, pasting, and reorganizing of terms, sub-terms and cross-references. While most of the PIs were pleased overall with the editing features of the present state of the art, the fact remains that they are still performing most of the intelligent portions of the process themselves. To follow up, we performed product testing of some of the current indexing software packages (CINDEX, MACREX, and SKY Index), and confirmed that they do not perform the intelligent/cognitive portions of the indexing process.

PIs are not the only individuals that an intelligent indexing tool would benefit. Informal interviews with academics that publish books and texts, such as English professors, show that many of them create their own indexes, and tend to use very primitive means (e.g. index cards or Microsoft Word) for generating these indexes. It's clear to us that academics, authors, teachers of library science and indexing, and anyone with an interest in organizing the key points of their books for the benefit of the user, or in educating students about such indexes, can benefit from the kind of time savings realized by the application that we are proposing.

There are numerous types of software available today for performing parts of the indexing process. Natural language parsers and taggers such as the Brill Tagger (Brill, 1993) determine which words or phrases are subjects. Concordance-generation packages, such as AntConc

(Anthony, 2006), use a cluster analysis technique to study the relationships between words and their surrounding text, but still require the user to provide specific words to search. Another application that uses cluster analysis, Grokker (Groxis, 2006), aids in searches of text by creating visual maps of related terms.

Many indexers use products that function like databases, and can be integrated with Microsoft Access and dBASE III, where each row of data consists of a heading, subheading (if applicable), and the page number (locator). I researched the trial versions of three such applications, which are the most popular among professional indexers, namely CINDEX (Indexing, 2006), Macrex (Macrex, 2005), and SKY Index (Sky, 2006). These programs take user-entered index information and provide an interface for such tasks as grouping, censoring, sorting (alphabetically or in some other order), searching and replacing terms, formatting and changing of type styles, arranging entries on the index page, type-ahead features to reduce redundant typing, spell-checking, and the ability to import and export index files.

All of these products focus on providing ways to reduce the mundane portions of the indexing process, so that the user can focus on the critical portions of the process, such as defining the terms to be indexed, and deciding which semantic relationships apply (e.g., which terms to mark as sub-terms, or as synonyms). None of these tools actually determine which terms are the correct parts of speech to index (i.e., nouns and noun phrases), nor do they attempt to capture context-sensitive information about the terms and their relationships to one another.

An early attempt to fully automate the indexing process, Indexicon, utilized the built-in index and tagging feature of Word Perfect and Microsoft Word, but failed to live up to product claims, as is detailed in a review of the product by Nancy Mulvany and Jessica Milstead (Mulvany, 1994). While Indexicon did allow the user to determine the *level of sensitivity* (i.e. a threshold for determining the level of significance of terms to include in the index) and *exclusion zones* (i.e. words or whole paragraphs to be excluded from the index), it failed to generate cross-references, and was found to omit many words that should have been included in the index, based on the user-selected criteria. Further, Mulvany found that Indexicon "suffer[ed] from a lack of completeness" and was "incapable of recognizing … terms that do not appear verbatim in the

text." Our prototype system uses mathematical techniques described below to identify synonyms for the purpose of providing cross-references, and disambiguates homographs into separate terms, using unique headings provided by the user.

Finally, I researched various mathematical and analytical techniques, finding that Latent Semantic Analysis (Landauer, 1998 and Wiemer-Hastings, 2004) / Latent Semantic Indexing (Deerwester, 1990) has demonstrated success in the field of text processing, where it's used to capture the semantic content of terms. LSA was found to perform at least as well as a second-language English speaker on tests of vocabulary and subject matter, can interpret metaphors, and is used in intelligent tutoring systems, grading essays, citing references, categorization of words, and information retrieval.

The three major works on Latent Semantic Analysis that I studied in preparation for this project are summarized below, where I have quoted the respective authors' accounts and summaries of this process. The mathematical details of the LSA process will be covered in detail in the EXPERIMENTAL DESIGN section of this paper.

In Peter Weimer-Hastings' paper, Latent Semantic Analysis (Weimer-Hastings, 2004), he describes LSA as "a technique for creating vector-based representations of texts which are claimed to capture their semantic content. The primary function of LSA is to compute the similarity of text pairs by comparing their vector representations. This relatively simple similarity metric has been situated within a psychological theory of text meaning and has been shown to closely match human capabilities on a variety of tasks."

Weimer-Hastings cites among LSA's benefits its "high correlation with human behavior", "the advantage of graceful degradation … [i]f it doesn't know a word, LSA simply ignores it and bases its representation on the other words."

He lists among its potential issues the fact that "the LSA approach … ignore[s] the syntactic structure of sentences … [and] ignores word order. In other words, LSA treats a text as a bag of words." Because of this, he cautions that LSA performs much better on larger text passages than

it does on individual sentences. For this reason, I implemented LSA in such a way as to group together paragraphs, rather than single sentences. There is a much greater probability of terms within a paragraph sharing similar contextual meaning with one another than they would with terms outside of the paragraph, thus strengthening the results.

In Thomas K. Landauer's paper, *An Introduction to Latent Semantic Analysis* (Landauer, 1998), he and his coauthors describe LSA as "a theory and method for extracting and representing contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other."

Landauer lists among LSA's strengths that "it mimics human word sorting and category judgments, it simulates word-word and passage-word lexical priming data; and … accurately estimates passage coherence, … and the quality and quantity of knowledge contained in an essay."

He cites among LSA's weaknesses the fact that "LSA … induces its representations of the meaning of words and passages from analysis of text alone. None of its knowledge comes directly from perceptual information about the physical world, from instinct, or from experiential intercourse with bodily functions, feelings and intentions. Thus its representation of reality is … somewhat sterile." LSA "makes no use of word order, thus of syntactic relations or logic, or of morphology."

Landauer notes that large corpuses have been experimentally shown to yield good results when reduced to between 50 and 400 dimensions. My application currently uses approximately 200 dimensions. I have proposed additional empirical studies, varying both the absolute number of reduced dimensions, as well as the number of reduced dimensions as a percentage of the original dimensions, to observe whether my results improve as a function of the number of dimensions in the reduced LSA matrices.

In Scott Deerwester's paper, *Indexing by Latent Semantic Analysis* (Deerwester, 1990), he and his coauthors describe Latent Semantic Indexing as an approach that "take[s] advantage of implicit higher-order structure in the association of terms with documents … in order to improve the detection of relevant documents on the basis of terms found in queries." Deerwester's Singular Value Decomposition technique (the factor-analysis portion of the LSA process), uses a reduced semantics space of 100 dimensions to approximate the original data matrix. What he refers to as *queries* "are represented as pseudo-document vectors formed from weighted combinations of terms." These queries return "documents with supra-threshold cosine values."

Deerwester is seeing promising preliminary results in his study, noting that LSI handles the synonymy (i.e., terms that are equivalent in meaning) problem quite well, although only partially solves the polysemy (i.e., term tokens that are diverse in meaning) problem: "It helps with multiple meanings because the meaning of a word can be conditioned not only by other words in the document but by other appropriate words in the query not used by the author of a particular relevant document. The failure comes in the fact that every term is represented as just one point in space. That is, a word with more than one entirely different meaning … is represented as a weighted average of the different meanings. If none of the real meanings is like the average meaning, this may create a serious distortion … [w]hat is needed is some way to detect the fact that a particular term has several distinct meanings and to subcategorize it and place it in several points in space."

I address this polysemy problem by using Word Sense Disambiguation. Rather than using the average context of a polysemous word, WSD looks instead at the contexts of the other terms surrounding each instance of a polysemous term (i.e., occurring in the same sentence or paragraph.) Within a certain distance threshold, we can group together instances whose surrounding text has a particular average context, and split apart those instances whose surrounding text has another average context. More details about this process can be found in the EXPERIMENTAL DESIGN section of this paper.

METHODOLOGY

My prototype software application uses Juola's (2005) six-stage approach to solve the problem of generating a back-of-the-book index.

In Stage 1, all potential index terms in the corpus are identified. The system takes advantage of existing natural language parsers and part-of-speech taggers to identify all nouns and noun phrases as good indexing candidates. A tagged version of the corpus is sent to the next stage of the system. In Stage 2, the system locates all informative references in the text. Stage 3 involves the identification and location of synonymous terms. Stages 2 and 3 are accomplished using singular value decomposition and factor analysis, based on the principles of Latent Semantic Analysis (Landauer, 1998 and Wiemer-Hastings, 2004) and Latent Semantic Indexing (Deerwester, 1990). The corpus is tagged / annotated to indicate relevance and synonyms. Stage 4 consists of the identification of index terms to split into sub-terms. Information from the previous stage will allow the system to determine which words always appeared near a given word or its synonym, which could possibly be sub-terms. This information will be tagged onto the corpus as well. Stage 5 is the development of cross-references within the index itself. The terms in the corpus will be annotated where these cross-references apply. Finally, Stage 6 is the compilation of page numbers and generation of the final index. This involves creating output in a format that an indexer can easily modify, but can also hand in to a publisher unedited.

EXPERIMENTAL DESIGN

My prototype machine-aided indexer uses several different techniques to achieve its end result. These processes are part-of-speech tagging, frequency analysis, Latent Semantic Analysis, Word Sense Disambiguation, Hierarchical Cluster Analysis, and clean-up and formatting procedures.

I will provide an overview of each of these processes, in the order in which they occur.

A. Part-Of-Speech (POS) Tagging

Words can be categorized into eight different "parts of speech." My software application, which is written in Java (chosen for its portability and platform independence), takes advantage of an existing natural language processor, MontyLingua (Liu, 2004), which tokenizes (i.e., normalizes punctuation, spacing and contractions), parses and tags the words in the raw text corpus with the appropriate parts of speech. Once all of the terms have been tagged with a delimiter and acronym, they are output from MontyLingua as tagged versions of the original text. The system can easily identify the nouns and noun phrases based on the tags provided. For example, *home* becomes *home/NN* since home is a noun. Part of speech tagging is of particular importance in indexing, as only nouns and noun phrases are typically included in an index (Chicago, 2003.) These nouns and noun phrases will become the starting point of potential terms to index.

MontyLingua, created by Hugo Liu of MIT's Media Lab (Liu, 2004), was chosen over other natural language processors (NLPs) and tagging tools that I evaluated, including the ANNIE information extraction software (University of Sheffield, 2005), NLParser (Charniak, 2005), LTChunk (Mikheev, 2000) and KBtextmaster (Watson, 2005), because it is a complete, end-to-end natural language processor, based on standards such as the Brill Tagger (Brill, 1993), and it incorporates common-sense knowledge, which makes it semantically rich. Another appealing and convenient feature is that it is available as a compiled Java library.

The following example illustrates raw text from Dickens' *Tale of Two Cities*, and the resulting MontyLingua tagged text:

Table 1 Raw Text vs. Tagged Text

Raw Text:
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us

Tagged Text:
It/PRP was/VBD the/DT best/JJS of/IN **times/NNS** ,/, it/PRP was/VBD the/DT worst/JJS of/IN **times/NNS** ,/, it/PRP was/VBD the/DT **age/NN** of/IN **wisdom/NN** ,/, it/PRP was/VBD the/DT **age/NN** of/IN **foolishness/NN** ,/, it/PRP was/VBD the/DT **epoch/NN** of/IN **belief/NN** ,/, it/PRP was/VBD the/DT **epoch/NN** of/IN **incredulity/NN** ,/, it/PRP was/VBD the/DT **season/NN** of/IN **Light/NNP** ,/, it/PRP was/VBD the/DT **season/NN** of/IN **Darkness/NNP** ,/, it/PRP was/VBD the/DT **spring/NN** of/IN **hope/NN** ,/, it/PRP was/VBD the/DT **winter/NN** of/IN **despair/NN** ,/, we/PRP had/VBD **everything/NN** before/IN us/PRP ,/, we/PRP had/VBD **nothing/NN** before/IN us/PRP

MontyLingua distinguishes between various types of nouns, using slightly different variants of the "/NN" acronym: /NN (singular noun), /NNS (plural noun), and /NNP (proper noun.) However, as far as my system is concerned, it merely suffices to know that a term is a noun.

Aside from its tokenizing and tagging capabilities, MontyLingua also has the ability to perform "chunking" (grouping together individual parts of speech in a sentence to form adjective, noun and verb chunks), and "lemmatizing" of raw text. Lemmatizing is the process of normalizing various forms of a word, such as tense or number, into one generalized version of the word, that can be used in an index (e.g., the lemmatized version of the words *sailing* and *sailed* is **sail**).

The initial prototype of this application only utilizes the parsing and tagging of nouns, but I plan to extend the model to include chunking and lemmatizing, to increase the accuracy and the flexibility of the model.

Some fundamental assumptions made by my application are that: (a) the human indexer will only be interested in indexing nouns and/or noun phrases, and (b) all of the terms input into the system are written in the English language. Although the system only uses nouns in its starting point of terms to index, the user will be able to include other terms in the final index. If the user wanted to index text in a language other than English, my application would need to use different natural language processing tools. Regardless of which NLP tool is used, it is important to note that we would still need to have one homogenous language throughout the text.

B. Frequency Analysis

Once the part-of-speech tagging has been performed, the system determines the frequencies of each of the terms, and sets upper and lower thresholds or frequency bounds so terms that appear either too frequently or too infrequently can be excluded from the list. The user can specify these bounds, otherwise pre-defined default values will be used by the system. The Chicago Manual of Style (Chicago, 2003) suggests that the number of terms in the index not exceed 5% of the total terms in the corpus.

C. Latent Semantic Analysis

This stage uses a factor analysis technique, based on the principles of Latent Semantic Analysis / Latent Semantic Indexing (Deerwester, 1990, Landauer, 1998, and Wiemer-Hastings, 2004), to study the relationships among the terms and their surrounding text, and generate numerical representations of the terms and their meanings. Factor analysis is defined as "a statistical technique used to explain variability among observed random variables in terms of fewer unobserved random variables called factors. The observed variables are modeled as linear combinations of the factors, plus 'error' terms" (Wikipedia, 2006). Within this application, the random variables are the dimensions of the matrices.

The supporting data in the LSA and LSI literature (Deerwester, 1990, Landauer, 1998, and Wiemer-Hastings, 2004) demonstrates their abilities to accurately represent and model human semantic knowledge. They are able to identify words that are similar in meaning and context without the terms having to be literal matches, to consistently identify synonyms and antonyms (which appear with a much higher degree of correlation than unrelated terms), and to show that compound words correlated extremely well with their respective component terms.

We begin the LSA process by taking a corpus and dividing it into subsections or *documents* (paragraphs), and analyze each unique word or *term* relative to each document. This portion of the analysis is specifically a *frequency* analysis. We determine the overall frequency of each term, as well as the frequency of each term within a given paragraph. We represent this information as a term-by-document matrix, whose rows represent the terms to index, and whose columns represent the documents in the corpus. Note that I chose to group the terms into paragraphs rather than individual sentences, because terms within a paragraph are likely to be more semantically related to one another than to terms outside of the paragraph.

Applying TF-IDF (*term frequency-inverse document frequency*) weightings to the values in the matrix, we can quantify the relative importance of each term with respect to each paragraph and to the corpus as a whole (in other words, how each term relates to each document.) This weighting equates to the number of times a term occurs in the corpus divided by the number of times it occurs in a given document. The logarithm of this value for each term is taken for scaling purposes. Next, these values are normalized (row by row) by taking the square roots of the sums of the squares of values across each row of the matrix, and dividing each cell's original result by the normalizing factor for that row. This normalization step is done to smooth out the data, minimizing the contributions of term instances that occur either very frequently or very infrequently in the document. The TF-IDF method, also known as "log entropy," can be thought of as the product of a value and its information gain (TF-IDF has its basis in Information Theory.)

Below is an example of a normalized term-by-document matrix for an excerpt of Bible text (ten of the most frequent terms in the excerpt, for the first ten documents.) We can see whether a term

occurs in a given document, how frequently it occurs in each document, and can compare the relative frequencies of terms.

Table 2 Term-by-Document Matrix (Excerpt)

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	Doc 9	Doc 10	Doc …
give	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2771	0.0000	0.0000	0.0000	…
light	0.6915	0.2744	0.0000	0.0000	0.5349	0.5879	0.0000	0.0000	0.0000	0.0000	…
beasts	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3431	0.0000	0.0000	…
division	0.0000	0.3451	0.0000	0.2589	0.0000	0.2464	0.0000	0.0000	0.0000	0.0000	…
ruler	0.0000	0.0000	0.0000	0.0000	0.4484	0.0000	0.0000	0.0000	0.0000	0.0000	…
signs	0.0000	0.0000	0.0000	0.0000	0.2066	0.0000	0.0000	0.0000	0.0000	0.0000	…
air	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5130	0.2157	…
bird	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3082	0.0000	0.0000	0.2399	…
arch	0.0000	0.7261	0.0000	0.2724	0.2359	0.5185	0.0000	0.0000	0.0000	0.0000	…
grass	0.0000	0.0000	0.4198	0.2724	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	…
…	…	…	…	…	…	…	…	…	…	…	…

A one-dimensional vector of term means is computed, whose length equals the number of unique terms in the corpus, and whose values represents the means of the values in each row of the term-by-document matrix (meaning all of the values of that term across all documents).

Table 3 One-Dimensional Vector of Term Means (Excerpt)

Term Mean
0.0065
0.0106
0.0052
0.0046
0.0052
0.0065
0.0069
0.0045
0.0046
0.0040
…

Next, a square term-by-term covariance matrix is created, in which the covariance* between every row term and every column term is determined (*in the case of cells along the diagonal, the covariance is equivalent to the standard variance, since the row and column terms are equal). The mean values from the vector of term means are used in our formulas for calculating (co)variance.

Covariance is defined as the amount by which two terms vary together (or alternately, the product of deviations of each term from its respective mean), and is expressed as $\boxed{Cov(x,y) = \Sigma((x - mean_x)*(y - mean_y))/N}$, where x and y are the two different terms, and N is the number of paragraphs.

Variance is defined as variability from the mean, and is expressed as $\boxed{Var(x) = \Sigma(x - mean_x)^2/N}$, where the numerator is summed over all terms x, and the denominator N equals the number of paragraphs.

Table 4 Term-by-Term Covariance Matrix (Excerpt)

Term	give	light	beasts	division	ruler	signs	air	bird	arch	grass	…
give	-0.000037	-0.000037	-0.000042	-0.000049	-0.000045	-0.000041	0.000127	-0.000043	-0.000044	-0.000041	…
light	0.000312	0.000312	-0.000069	-0.000080	-0.000074	-0.000067	0.000115	-0.000071	-0.000072	-0.000067	…
beasts	-0.000030	-0.000030	-0.000034	0.000128	-0.000037	-0.000033	-0.000033	-0.000035	0.000132	-0.000033	…
division	-0.000026	-0.000026	-0.000030	-0.000034	-0.000032	-0.000029	-0.000029	-0.000031	-0.000031	-0.000029	…
ruler	-0.000030	-0.000030	-0.000034	-0.000039	0.000200	-0.000033	-0.000033	0.000224	-0.000036	-0.000033	…
signs	-0.000038	-0.000038	0.000053	0.000532	-0.000045	-0.000041	-0.000041	-0.000044	-0.000044	0.000130	…
air	-0.000040	-0.000040	-0.000045	0.000150	-0.000048	-0.000044	-0.000044	-0.000046	-0.000047	-0.000044	…
bird	-0.000026	-0.000026	-0.000029	-0.000034	-0.000031	-0.000028	-0.000029	-0.000030	-0.000030	-0.000028	…
arch	-0.000026	-0.000026	-0.000030	-0.000034	-0.000032	-0.000029	-0.000029	-0.000031	-0.000031	-0.000029	…
grass	-0.000023	-0.000023	-0.000026	-0.000030	0.000208	-0.000025	-0.000026	-0.000027	-0.000027	-0.000025	…
…	…	…	…	…	…	…	…	…	…	…	…

It is the term-by-term covariance matrix that provides the input for the next stage of the application.

This stage is a type of multivariate statistical analysis used by LSA / LSI, known as Singular Value Decomposition (SVD). This is a type of factor analysis, which is used to explain variability among random variables (terms in the corpus represented in an n-dimensional semantic space) in terms of a reduced number of random variables (less dimensions).

The SVD process is used by the application to perform a decomposition of the covariance matrix into three component matrices (one describing the row values as eigenvectors of orthogonal factors, another describing the column values as eigenvectors of orthogonal factors, and a diagonal scaling matrix of eigenvalues). One can think of eigenvectors as scaled-down

transformations of the row and column data, and eigenvalues as the magnitude by which they have been scaled (i.e., the scaling factor). The resulting eigenvalues are ranked and sorted from largest (most significant) to smallest (least significant) in absolute value.

I utilized a third-party package created by the National Institute of Standards and Technology, or NIST, called JAMA (*JA*va *MA*trix Package) to perform the SVD methods (NIST, 2006).

The mathematical details of SVD are described by Deerwester (1990): in SVD, any rectangular (and trivially, any square) matrix X can be decomposed into the products of three separate matrices, T_0, S_0, and D'_0, such that $\boxed{X = T_0 S_0 D'_0}$

These three resulting matrices have the following properties:

T_0 (term matrix) and D_0 (document matrix) have columns that are *orthonormal*, which means (mathematically) that they are perpendicular and of unit length, and which also means (statistically) that the variables (or in our case, the terms) are independent from one another. S_0 (singular matrix) is *diagonal* (meaning all values not lying along the diagonal are zero.) D'_0 is the *transpose* of the matrix D_0, meaning that the rows of D_0 become the columns of D'_0.

The decomposition of matrix X occurs as follows (Wikipedia, 2006): The dot product of each pair of terms gives us the correlation between the terms relative to the documents. The matrix product $X X^T$ contains all such dot products (this is T_0). Similarly, the dot product of each pair of documents gives us the correlation between the documents relative to the terms. The matrix product $X^T X$ contains all such dot products (this is D'_0). With this fact about X in mind, we can see what it means for X to be decomposed into the two orthonormal matrices T_0 and D'_0 and the singular matrix S_0 described above. This action is the actual "singular value decomposition."

To elaborate, the matrix D_0 represents the documents in n-dimensional space (where n is a large number, on the order of 10^2 to 10^3), the matrix T_0 represents the terms in n-dimensional space, and the matrix S_0 is a zero-filled matrix containing non-zero "singular" values along its diagonal.

These singular values are also known as *eigenvalues*, and the rows of data in T_0 and D'_0 are known as *eigenvectors*.

The eigenvalue matrix is a one-dimensional array consisting of factors by which our covariance matrix relates to the resulting eigenvector matrix. If we rank or rearrange these eigenvalues from largest to smallest (in absolute value), we are able to re-sort the rows of the corresponding eigenvector matrix as well, so that the resulting term eigenvector matrix T_0 is prioritized in order of importance. The eigenvector matrix T_0 gives us a least-squares fit of all of the dimensions of the problem (how terms relate to other terms). These dimensions are in a coordinate system defined by the eigenvalue space. We need to transform them to dimensions in our corpus's term-document space, so we take the dot-product of each row of our covariance matrix with each column of the sorted eigenvector matrix T_0 in turn. The result is a new matrix, which we call the coordinate matrix, as it contains all of the coordinates, or dimensions, of our problem in the space of our corpus.

The key to LSA's success is its ability to take the complete term and document information, in the n-dimensional space, and based on the information provided by the ranked eigenvalues and corresponding eigenvectors, remove the least important dimensions from that space, and produce a smaller, less noisy representation of the terms and documents in a new k-dimensional space sharpening any similarities and/or constrasts within the relationships among the terms.

This reduced, reconstructed data shows us how the terms correlate with one another in the semantic space. The proximity of terms to other terms allows us to make educated inferences as to which terms appear to be synonyms (correlating highly in all dimensions) and which terms may be subcategories of broader terms (correlating highly in one or more dimensions.) This ability to determine term relationships mathematically provides our system with its intelligent *context-sensitive* nature.

The literature on LSA suggests that there exists an optimal number of dimensions that is less than the total number of dimensions in the problem space. Experiments on large corpuses showed ranges of 100 to 400 dimensions as optimal. Because SVD allows us to control the

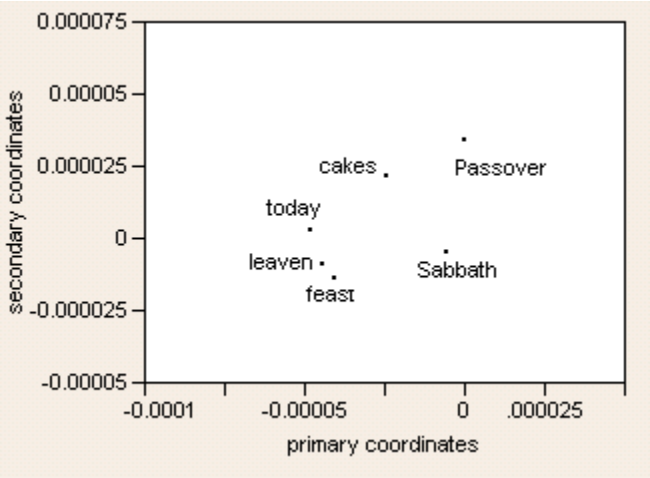
number of dimensions, a configurable threshold is set in the prototype application to adjust the model if the clusters of terms appear to be too restrictive or too unrestrictive in the semantic space. Thus, we are able to specify what values are considered to be "correlated enough" in each dimension to be considered as related terms. I have set the system's default number of reduced dimensions to be 200. Note that an inexperienced user may feel more comfortable using the default "number of dimensions" setting, as well as the defaults for any other configurable settings provided by the system, rather than endeavoring to change them.

The table below provides a subset of Bible data from LSA's resulting coordinate matrix, and the figure following it depicts a scatter plot of these terms in the first two (most significant) dimensions. Note that these particular terms were selected for illustration because they lie near to each other in the semantic space, although they may not have occurred in the first ten documents of the corpus, as did the data shown in the previous tables.

Table 5 Coordinate Matrix Data (Excerpt)

Term	Primary Dimension	Secondary Dimension
feast	-0.000040	-0.000014
today	-0.000048	0.000003
Sabbath	-0.000006	-0.000005
cakes	-0.000024	0.000021
Passover	0.000000	0.000034
leaven	-0.000044	-0.000009

Figure 1 2D Plot from Coordinate Matrix



To reiterate, the preceding figure merely depicts a slice of the semantic space in two dimensions. Now, imagine a graph of points for these same terms in 200 dimensions. The system has identified the 200 most significant dimensions and used them to pinpoint a location in 200-dimensional "meaning space" corresponding to the meaning of each term type. The closer that two or more points are to one another, the more similar their semantic meanings. We can observe that words of similar meaning (such as synonyms and sub-terms) exist in close proximity in this multidimensional space.

LSA enables us to strip away unnecessary information (noise) to create a clearer picture of the terms and their relationships to one another. Because LSA is completely dependent on the terms contained in the corpus, and does not rely upon dictionaries or thesauri for supplemental information, it is able to more accurately reflect the subtle shades of meanings of terms that may be unique to a particular trade or field of study that is being described within the corpus.

D. Word Sense Disambiguation

After all of the relationships among terms have been identified by LSA, producing numerical encodings that give the average context for each term token, the system performs Word Sense Disambiguation (WSD) to differentiate among term tokens that are spelled identically but have different meanings (e.g., homographs or polysemous terms), categorizing them based on what "sense" or meaning of the term is being used.

For example, suppose the word "bass" is used throughout a corpus, and that it is used in some parts of the corpus to refer to a type of fish, but is used elsewhere to refer to a musical instrument. If we would be satisfied with the numerical encodings of context generated by LSA, we would have one average encoding for the word "bass" – however, this kind of "average context" really doesn't make sense for this particular word. Therefore, we need to examine each occurrence of "bass" in closer detail, looking at the words immediately surrounding it, and applying WSD to clarify, or disambiguate, it into, say, bass_fish and bass_instrument.

The WSD technique is implemented in the following manner:

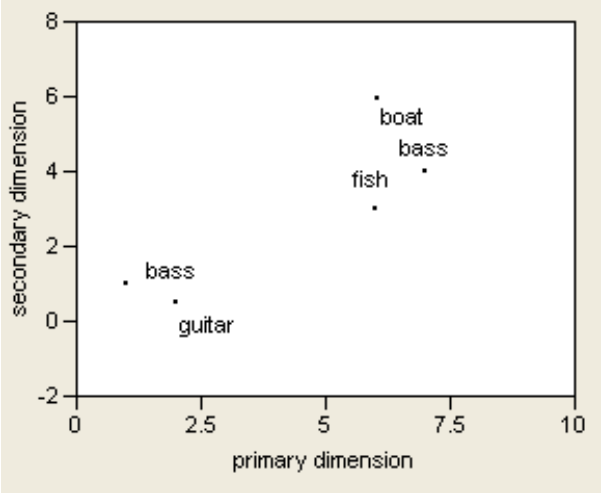
For each term type, we need to determine whether it is used in more than one context. Whereas LSA gives us one average context representation per term type, WSD disambiguates each term type into all contexts used in the corpus. For a given term type, we compute this by taking the average of the type semantics of each of the remaining terms in a sentence, which we can leverage from our LSA data. Plotting these coordinates for all contexts of a given term type, we can observe how closely the points do or do not cluster.

We can determine the distance between any two terms ("a" and "b") by using the Euclidean distance measure, which takes the square root of the sums of squares of the distances between the terms in each dimension. For example, for two terms, each represented in n dimensions, we express their Euclidean distance from one another as $\sqrt{[(a_1-b_1)^2+(a_2-b_2)^2+\ldots+(a_n-b_n)^2]}$ or more compactly as $\sqrt{\Sigma_i(a_n-b_n)^2}$, where i ranges from 1 to n.

We also set a distance threshold in our system (which may be overridden by an experienced user) to determine how close terms need to be before they are considered to be of the same context. In other words, if a Euclidean distance measure is within (i.e., less than or equal to) this distance threshold, the pair of terms is considered to have the same sense. If the Euclidean distance is outside of (i.e., greater than) this distance threshold, then the terms are considered to have different senses. All pairs whose Euclidean distances are close enough to each other but farther from the others will cluster under the same meaning. Thus can we discern between senses/contexts for polysemous terms.

In our example, instances of "bass" whose surrounding text is within this threshold of one another are grouped under one heading, and instances of "bass" whose surrounding text is outside of the threshold but are close to each other are grouped under another heading, and so on. We can obtain from the user the keywords used to differentiate these in the system (e.g., "_fish" and "_instrument"), but in the absence of this user input, the system will provide more generic differentiations (e.g., "_A" and "_B"). Figure 2 illustrates "bass" as a polysemous term, disambiguated.

Figure 2 Example of "Bass" as a Polysemous Term



We can see by plotting the first 2 dimensions of the semantic data associated with each instance of "bass" and its surrounding terms that the different meanings of the term naturally cluster into two separate groupings.

WSD provides for us a level of granularity and robustness that is needed to generate an intelligent first draft index, by capturing this representation of real world knowledge that has previously been performed, out of necessity, by the human indexer.

E. Hierarchical Cluster Analysis

Once all terms have been encoded with their respective semantic meanings, and the system has disambiguated multiple occurrences of polysemous terms, the Hierarchical Cluster Analysis (HCA) stage begins.

HCA describes a method used to partition terms into subsets with similar properties or characteristics. Members of an optimally clustered group share maximum characteristics with one another and minimal characteristics with terms in any other group, so terms that are semantically similar will cluster together. In the context of this application, this information can be used to identify synonyms and sub-terms, by iterating back through the WSD and HCA processes as many times as is needed. Note that antonyms as well as synonyms will cluster

together, as these both have strong relationships, and the system cannot distinguish whether these relationships are positive or negative.

The Chicago Manual of Style (Chicago, 2003) has set a guideline that, if a term has more than 5 or 6 individual references in the corpus, it should be clustered into broader categories in the index. This is what HCA accomplishes for us, as it merges these terms together into clusters of references rather than individual references, and gives user-provided names to the generalized categories. We also have the ability to reintroduce merged terms as sub-terms under the more generalized categories.

Each pass through the HCA algorithm is a partitioning step -- terms that are closer to one another (as determined by the distance threshold) than to those in any other groups are clustered together.

Before implementing the algorithm, the system prompts the user (or uses a default setting) for thresholds of between-term and between-cluster distances to use as cutoffs for similarity.

Pass 1

For the first pass through the algorithm, each term begins in its own cluster. There are various ways of measuring distance between terms. I chose to use Euclidean distance, which takes the square roots of the sums of squares of the distances between the sets of terms in each dimension. These distance values are based on the coordinates of each term, as determined by the LSA process. The Euclidean distances are stored in a matrix, sorted from closest to farthest. Terms that lie within the distance threshold are clustered together.

Pass 2 through n – 1

For the next several passes, the system determines the average distance between each pair of clusters. I chose to use the Centroid distance measure for the comparisons. This distance measure calculates the point in the center of each cluster (in each dimension), by determining the midpoint in the distance between each pair of points in the cluster, and taking the intersection of all of these midpoints to determine the center ("center of mass") of the cluster. These values are

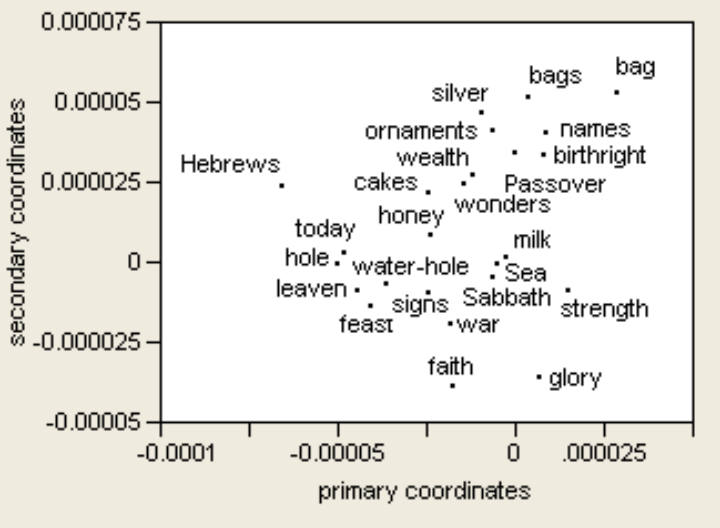
stored in a matrix, sorted from closest to farthest. Clusters within the distance threshold of one another are grouped together.

Pass n

This process continues until no groups remain that can be further clustered (all remaining clusters that fall within the threshold distance are clustered together.) The user is able to give generalized names to clusters of terms (so that the terms can be listed as sub-terms under this more generalized heading.)

A subset of data run on a passage from the Bible is represented as a plot of the first two (most significant) dimensions of data (see Figure 3 below).

Figure 3 2D Plot of Bible Data



There is also another way of viewing the clustering of the data. A hierarchical cluster, or tree cluster, is most commonly represented by a dendrogram chart, which illustrate the treelike structure of the clusters of data: all of the unique terms are leaves of the tree, and moving up the tree towards the top, similar terms are assigned to the same branch, and similar branches are assigned to the same larger branch closer to the top of the tree, and so on, until the top of the tree is reached. Mathematically, as terms are branched together, their individual values are replaced by weighted, scaled values that represent the clusters of terms on each branch collectively.

Figures 4 and 5 show two views of a dendrogram representation of the same subset of Bible data:

Figure 4 Dendrogram of Bible Data

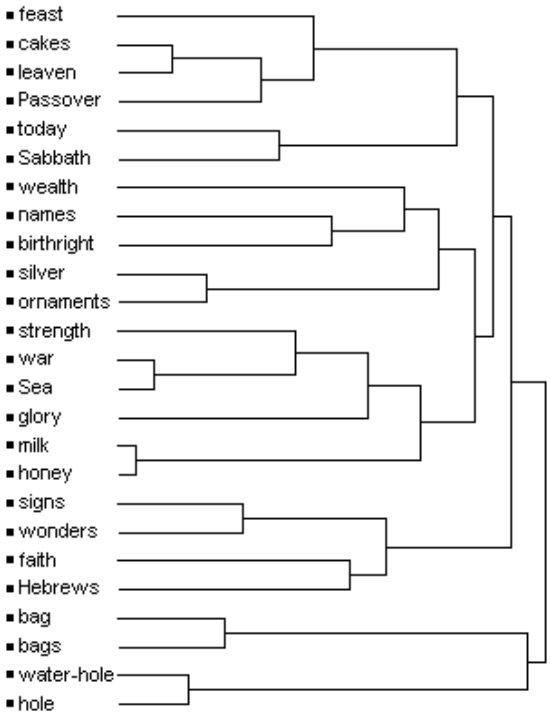
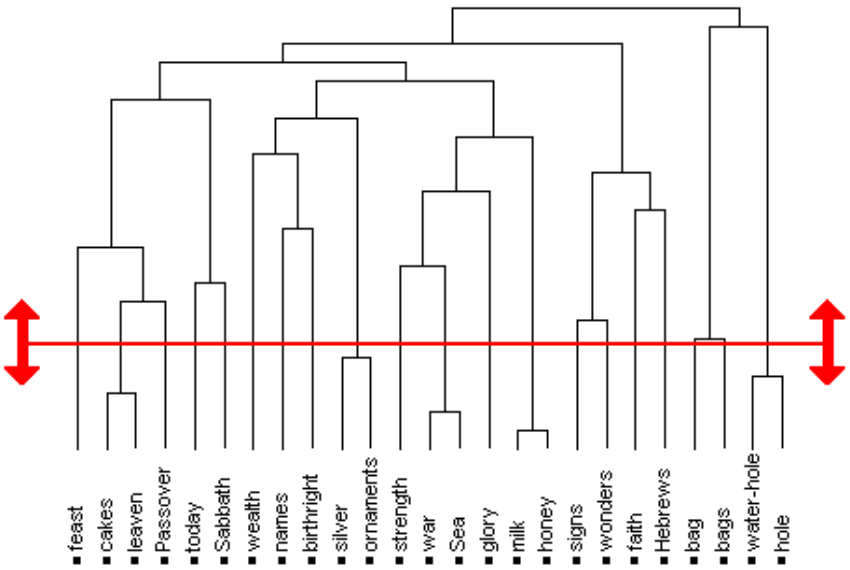


Figure 5 Dendrogram of Bible Data with Distance Threshold "Slider"



The dendrogram begins with all terms starting in their own groups, and are continually grouped together until they can be grouped no longer. The configurable distance threshold set by system (or overridden by an experienced user) determines how many groups we end up with (or, in other words, when to stop clustering). We don't want each term in its own group, but we also don't want all terms in one group. Thus, the optimal clustering distance lies somewhere between the two extremes. The key is to choose distance thresholds that maximizes similarities and minimize differences among the terms, optimizing representations of the term relationships. (The bold "slider" line on the chart represents the ability to slide up or down to find the optimal distance threshold.)

As we compare the two charts, it is not surprising that terms that were grouped together in scatter plot also appear grouped together in dendrogram. (e.g., "milk" and "honey" are relatively close to one another, and neither is as close to "bag" or "bags").

HCA is an extremely powerful data analysis tool, and in our particular application, it allows us to identify words that have a high degree of correlation after LSA and WSD (translating into similar contextual meanings), and can therefore be included in the index as either synonyms (i.e., *see also* or *refer to* values) or sub-terms (i.e., indented terms listed beneath a more general term in the index).

F. Formatting of Final Index

As terms are disambiguated in the WSD step, the original text is tagged with delimiters to indicate whether the term has one particular sense or another. For example, each instance of "bass" will be delimited as either bass/WSD_fish or bass/WSD_instrument, where appropriate.

As terms are clustered together in the HCA step, the user is given the opportunity to declare them as synonyms, or to assign a more general term (so that they will appear beneath it as sub-terms.) Depending on which case occurs, the original text will be tagged with the appropriate delimiter. For example, clustered terms such as "signs" and "wonders" from the Bible example could be assigned a more general category of "miracles", so that the terms are delimited as

signs/HCA1_miracles and wonders/HCA1_miracles. In the case of synonyms, such as "hole" and "water-hole" in the Bible example, one would be chosen to be the main index term, and the others the "see also" terms: thus, "water-hole" would be delimited as water-hole/HCA2_hole.

Note that this preliminary prototype reads in plain text, which does not explicitly handle page numbers. For this reason, I have stored pseudo-page-numbers for each term that is read in, in order to demonstrate how the system works. These page numbers were generated based on an average of 300 terms per page (determined experimentally by averaging the number of terms on each page of a typical text file.) One of my top priorities is to incorporate the reading of XML tags into my system, so that proper page number information can be associated with each term.

After all of the relationships among terms have been identified and refined (disambiguated and clustered together, possibly several times iteratively), and the appropriate delimiters added to the terms in the original text, the system generates a structured hierarchical index, alphabetized by main terms, subcategorized where appropriate, listing cross-references where applicable, and enumerating the page numbers corresponding to these terms (see Figure 6 below).

Figure 6 Excerpt from Final Index

bass, fish	3, 19, 27
bass, instrument	48, 142
hole	12 - 15
miracles	
signs	78-80
wonders	79
water-hole (*see also, hole*)	37

DISCUSSION

A. Machine-Aided System

I would like to emphasize that this prototype system is machine-*aided*, as opposed to *fully* machine-automated. The purpose of a machine-aided automatic indexer, and its distinction from a fully-automated indexer, is to assist the human indexer in his or her work, but *not* to replace the human, whose judgment and experience cannot be fully captured by even the most sophisticated expert system.

The user will be able to edit the results of the automatically generated index at any point in the process, as well as perform post-editing of the final output, and will at any time be able to re-include terms that were previously excluded from the index. In addition, it is the user's responsibility to specify a more general term to use for a collection of synonymous terms.

To summarize the process (in which all stages can be modified by user input and refinement):

- A user will input a corpus into the system.
- The system will parse and tag all terms in the corpus, and perform a frequency analysis of the terms – this will be the first level of term filtering in the index.
- The system will perform LSA to determine the context of each unique word in the corpus.
- The system will provide WSD to check for any polysemous terms, and will provide an opportunity for the user to clarify these as well.
- The system will furthermore use HCA to group together related terms, and will provide an opportunity for the user to refine these groupings.
- The system will generate a final marked up version of the corpus containing embedded index tags including page references, and from this will also generate the template for the index pages themselves.

To distinguish between a machine-*aided* automatic indexer and a fully automated indexer, it is helpful to draw an analogy to the differences between machine-aided translation and full machine translation.

Full machine translation (MT) is a process by which a computer algorithm determines the context of the terms or phrases in one language and translates them into corresponding terms or phrases in another language. One method of machine translation, described by Juola (1989), uses the *interlingua* approach, in which "the meaning and structure of the internal representation are independent of the language from which they are derived." This approach approximates a "true 'machine understanding' of language." The main point I wish to make about MT is that it's a method in which the machine handles all of the translation process. In other words, all intelligent tasks are performed by the machine.

Conversely, with machine-aided translation (MAT), also known as computer-assisted translation, the system's function is to supplement or assist the human translator, rather than performing the actual translations from one language to another. Examples of MATs include spell checkers and grammar checkers. In contrast with MT, the intelligent tasks are still performed by the end user (e.g., the user chooses the words in the second language, and the grammar checker tells the user if the chosen words form a grammatically complete sentence.)

Wikipedia's (2006) definition of the machine-aided translation process in the following way: "the computer program supports the translator, who translates the text himself, making all the essential decisions involved." In contrast, it describes the full machine translation process in the following manner: "the translator supports the machine, that is to say that the computer or program translates the text, which is then edited by the translator, or not edited at all."

The parallels with machine-aided and full machine indexing are obvious. My prototype system, while performing some of the intelligent tasks, does not presume to perform all of them at the same level of ability as would an experienced human indexer. The application aims to provide the best starting point for the user, but provide several means throughout the system for the user

to modify the index as it is being created. All final decisions are to be made by the human indexer before the index is accepted as final.

To reiterate, this prototype's goal is to greatly reduce the time required for the human indexer to perform the indexing process, while simultaneously delivering the best first draft at a back-of-the-book index. By making informed suggestions, our application provides great value, but it also allows the user to incorporate a component of real-world knowledge about the subject matter that is extremely difficult to model.

B. Results and Testing

The task of evaluating the prototype application's overall performance (i.e., how "close" the resulting indices are to fulfilling the guidelines described in this paper) is a difficult one, which I will attempt to answer by performing side-by-side comparisons of an index generated for a given corpus by our application and by a typical human indexer. By comparing the indexes generated by each, and create a matrix comparing (i) what my application and the indexer both included in the index, (ii) what my application included but the indexer did not, (iii) what the indexer included but my application did not, and (iv) what neither one included. I will also note the relative times required for performing the task in each case. After quantifying what "percentage of agreement" between my application's results and that of the human indexer is acceptable, I believe that if this prototype application can generate a similar index in a fraction of the time, then it has proven itself to be a success.

As mentioned earlier in this paper, many metrics exist for determining this system's success or failure, including the guidelines outlined by the National Information Standards Organization (NISO) and the Chicago Manual of Style. In fact, there are many authoritative resources available that provide guidelines for defining a "good" index. There is certainly room for substantial research in studying this system and comparing it with the metrics identified in these guidelines, and evaluating this tool to determine if it has met all of the guidelines. If I fall short in any of them, I will revise appropriate elements of the system, and continue to re-evaluate the system.

While this prototype is still a work in progress, I am seeing significant results so far, using a combination of Latent Semantic Analysis, Word Sense Disambiguation and Hierarchical Cluster Analysis. I am able to locate all of the potentially informative terms in the body of text, namely nouns and noun phrases, and allow the user to set the thresholds for how frequently a term must occur in order for it to be included in the index.

Performing LSA and subsequent HCA shows the first and most restrictive level of clustering to be nearly 40% accurate upon inspection. This is based on comparisons between the terms that the system clustered together and my own real-world knowledge of whether these terms are actually related and should actually have been clustered together.

Reading a corpus into the current PC-based prototype, which has approximately 60,000 words in 600 paragraphs, takes several hours to process. By nature, the LSA algorithm requires a tremendous amount of computation in the form of manipulations of matrices in several thousand dimensions. This has proven to be a formidable task for a personal computer. However, in spite of this, I am seeing significant clusters of terms. Statistically, larger corpora will certainly generate better results, as they provide a more complete dictionary of the language of the corpus for the system to process. To circumvent the performance limitations of running a large-scale matrix analysis on a PC, Dr. Juola and I hope to take our research to Duquesne University's cluster of large-scale supercomputers, pending approval of our NEH funding. By doing so, we will be able to increase performance, allowing us to handle much larger corpora, thus yielding better statistical results. Of course, even a system that takes hours to run is still preferable to a totally manual process that takes weeks to complete.

At the time of publication, I am continuing to refine all of the parameters in the model to more closely achieve an index that matches that of a human indexer. I recognize that there are additional improvements and refinements to be made, but am encouraged by the preliminary results.

To that end, I have composed a list of open research questions and possible empirical studies, that will be included in future extensions of this project:

1. What are the optimal minimum and maximum frequencies of terms that should be included in a back-of-the-book index?
2. Which TF-IDF weighting method is best for the term-by-document matrix?
3. What is the optimal number of dimensions (or percentage of the overall dimensions) for achieving the best resulting LSA matrix?
4. What are the optimal semantic distances between terms in order for them to be considered synonyms, sub-terms, or distinct separate terms?
5. What methods for determining between-terms distance measure (e.g. Squared Euclidean, Manhattan, Chebychev, Power, and Percent Disagreement) provide the best representation of the data?
6. What methods for determining between-clusters distance measure (e.g. Nearest Neighbor, Furthest Neighbor, Unweighted pair-group average, Weighted pair-group average, Unweighted pair-group centroid, and Ward method, which uses an analysis of variance to minimize error between clusters) provide the best representation of the data?
7. What is the optimal Percentage of Agreement of the prototype's index compared with that of an index generated completely manually?

As each of these parameters is varied, the goal is to find the optimal values, which will provide the best draft index.

POTENTIAL APPLICATIONS

Dr. Juola and I anticipate that this system will be subject to incremental refinement, as we research ways of increasing the accuracy of the generated indexes. With that in mind, we have designed the architecture of our prototype system to be modular in nature, easily facilitating the replacing of individual components without impacting other areas of the system. Future implementations will continue to incorporate the most efficient indexing and retrieval methods.

When the prototype is sufficiently accurate, we are planning a second phase in which to extend and enhance the system to make it more widely useful and available.

First and foremost on the list of enhancements, the system needs a robust, effective user interface that conforms to guidelines for good human-computer interaction and interface design. This interface should provide a screen showing the user all of the words that have been excluded from the index, as well as relationships among all included terms, and should also duplicate all of the mundane formatting / searching features that are offered by current state-of-the-art applications.

Other proposed enhancements include *productizing* the system to read in and format index output in a variety of word processing and searching platforms, including Microsoft Word, Adobe PDF, LaTeX, and XML, which are all widely used by various groups within the digital humanities community. Of these, the most popular request by far has been for the system to work with XML formatting. We would also like to *internationalize* the system, making it extensible to non-English languages.

We plan to research and implement an extension of this application to include not only written and electronic text, but also audio resources, such as audio books and recorded lectures. This would allow users to easily derive the content of such audio files, and to quickly search and navigate to the part of the audio recording that is relevant to the user. It may also be beneficial to consider indexing metadata or header information pertaining to graphical data. By extending to other forms of media, the application serves to provide and enhance intellectual access to humanities resources.

We are also considering future extensions of our system that may include supplementing the data in the LSA matrices with known word space vectors for those terms, based on dictionaries or thesauri, in order to further reduce noise and misclassifications, and strengthen any existing relationships among terms.

We plan to investigate other methods of numerically encoding the context of words (besides Latent Semantic Analysis) for comparison. Among other techniques available (currently being used in other areas of humanities computing) is Pointwise Mutual Information & Information Retrieval (PMI-IR), which is an unsupervised learning algorithm for recognizing synonyms, and is used for querying web search engines (Turney, 2001). Another example is WordNet, which is an online lexical reference system developed at Princeton University (Princeton, 2005).

I foresee the use of this intelligent, machine-aided indexer as an aid for professional and non-professional indexers alike, and believe it has great potential as a teaching tool. It can be augmented to be more specific to fields of study having their own niche sets of word semantics.

There is great work being done in the field of library science on ways to improve indexing and information retrieval techniques, and we believe that our application would be useful in helping to develop these skills.

In the field of linguistics, our application may be used to aid in the teaching of synonym detection, and to study how words vary between disciplines. Our tool is naturally a fit because of LSA's ability to successfully determine the contexts of industry- or discipline-specific terms, based purely on their locations within the corpus.

We also envision our application being used to quickly summarize and categorize large quantities of legal text to aid in the discovery process, or to summarize large quantities of medical text where often several terms in Latin, colloquial English, etc. are used to represent the same affliction, so that a glossary of medical terms can be quickly generated.

CONCLUSIONS

To summarize, this prototype back-of-the-book indexing tool is machine-aided, supplementing but not replacing the user, in which the user can still make the final decisions in the process. Intelligence has been incorporated into the system to perform the major cognitive tasks of the indexing process, providing a clear advantage over current state-of-the-art tools. This will significantly reduce the time required for creating the draft index, thus reducing this burden on the human indexer. Additionally, extensions of this system into other applications of text analysis are currently being explored.

I believe that the framework that Dr. Juola and I have incorporated into this prototype system takes us well on our way to meeting the challenge that was stated at the beginning of this paper, namely to develop a software program that bridges the gap between computerized concordances and manual indexing, to provide a much more robust draft index, which a human indexer can refine in a fraction of the time.

REFERENCES

Anthony, L. (2005). *AntConc 3.1.2 concordance generation software*, http://www.antlab.sci.waseda.ac.jp (accessed 22 October 2005).

American Society of Indexers. (2005). http://www.asindexing.org/site/index.html (accessed 22 October 2005).

Anderson, J. (1997). *NISO Technical Report 2: Guidelines for Indexes and Related Information Retrieval Devices*. Bethesda: NISO Press, p. 8.

Brill, Eric. (1993). *Brill Tagger software*, http://www.cs.jhu.edu/~brill (accessed 27 November 2005).

Brill, Eric. (1993). *A Corpus-Based Approach to Language Learning.* Doctoral dissertation, University of Pennsylvania.

Charniak, Eugene. (2005). *NLParser software*, ftp://ftp.cs.brown.edu/pub/nlparser (accessed 27 November 2005).

Cunningham, H., et al. (2002). "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications." *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02).*

Deerwester, S., et al. (1990). "Indexing by latent semantic analysis." *Journal of the American Society for Information Science,* 41(6): 391-407.

Groxis, Inc. (2005). *Grokker software*, San Francisco, CA. http://www.groxis.com (accessed 22 October 2005).

Indexing Research. (2005). *CINDEX Indexing Software*, New York, NY. http://www.indexres.com (accessed 22 October 2005).

Juola, P. (1989). "Machine Translation and Lojban." *Ju'i Lobypli.* 8.

Juola, P. (2005). "Towards an Automatic Index Generation Tool." *Proceedings of the 2005 Joint Annual Conference of the Association for Computing and the Humanities and the Association for Literary and Linguistic Computing (ACH/ALLC 2005).*

Juola, P. (2006). "Killer Applications in Digital Humanities." Submitted to *Proceedings of the 2006 Joint Annual Conference of the Association for Computing and the Humanities and the Association for Literary and Linguistic Computing (ACH/ALLC 2006).*

Jurafsky, D., Martin, J. (2000). "Word Sense Disambiguation and Information Retrieval." *Speech and Language Processing*. NJ: Prentice Hall, pp. 631 – 666.

Landauer, T., et al. (1998). "An Introduction to Latent Semantic Analysis." *Discourse Processes*, *25*: 259 – 284. Mahwah, NJ: Erlbaum Associates.

Liu, H., MIT Media Lab. (2004). *MontyLingua: An end-to-end natural language processor with common sense.* http://web.media.mit.edu/~hugo/montylingua (accessed 12 February 2006).

Lukon, S., Juola, P. (2006). "A Context-Sensitive Machine-Aided Index Generator." *Proceedings of the Joint Annual Conference of the Association for Computing and the Humanities and the Association for Literary and Linguistic Computing (ACH/ALLC 2006)*. University of Paris-Sorbonne. July 5, 2006: pp. 327-328.

Lukon, S., Juola, P. (2006). "Designing a Context-Sensitive Machine-Aided Index Generator." *Proceedings of the Annual Conference of the Canadian Symposium on Text Analysis Research (CaSTA 2006)*. University of New Brunswick. October 11, 2006: pp. 115 - 125.

Macrex Indexing Services. (2005). *Macrex Indexing Program*, Daly City, CA. http://www.macrex.com (accessed 22 October 2005).

Mikheev, Andrei. (2000). *LTChunk V 3.0 software*, http://www.ltg.ed.ac.uk/software/pos (accessed 27 November 2005).

Mulvany, N., Milstead, J. (1994) "Indexicon, The Only Fully Automatic Indexer: A Review," http://www.bayside-indexing.com/idxcon.htm (accessed 27 June 2006).

National Institute of Standards and Technology (NIST). (2006). JAMA: Java Matrix Package. http://www.math.nist.gov/javanumerics/jama (accessed 4 May 2006).

O'Grady, W., et al. 2001. "Computational Linguistics." *Contemporary Linguistics*, *4th Edition*, 663 – 703. Boston: Bedford/St. Martin's.

Press, W., et al. (1992). "Singular Value Decomposition." *Numerical Recipes in C: The Art of Scientific Computing*, 59 – 70. Cambridge: Cambridge University Press.

Princeton University. (2005). *WordNet 2.1 online lexical reference system,* http://wordnet.princeton.edu (accessed 12 October 2006).

SKY Software. (2005). SKY Index 6.0 Professional Edition, Stephens City, VA. http://www.sky-software.com (accessed 22 October 2005).

Smith, L. (2002). "A Tutorial on Principal Components Analysis." http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf (accessed 17 January 2006).

Turney, P. (2001). "Mining the web for synonyms: PMR-IR versus LSA on TOEFL." *European Conference on Machine Learning*, pp. 491-502.

University of Chicago Press Staff. (2003). *The Chicago Manual of Style*, *15th Edition*, Chicago: University of Chicago Press.

University of Sheffield. (2005). *ANNIE Information Extraction software*, http://www.aktors.org/technologies/annie (accessed 27 November 2005).

Watson, Mark. (2005). *KBtextmaster Version 2.0 software*, http://www.markwatson.com/opensource (accessed 27 November 2005).

Wiemer-Hastings, P. (2004). "Latent Semantic Analysis." *Encyclopedia of Language and Linguistics***.** Oxford: Elsevier.

Wikipedia.org. (2006). "Computer-assisted translation." http://en.wikipedia.org/wiki/Computer-assisted_translation (accessed 11 September 2006).

Wikipedia.org. (2006). "Factor Analysis." http://en.wikipedia.org/wiki/Factor_analysis (accessed 14 August 2006).

Wikipedia.org. (2006). "Latent Semantic Analysis." http://en.wikipedia.org/wiki/Latent_semantic_analysis (accessed 14 August 2006).