Adapting LSI for Fine-Grained and Multi-Level Document Comparison

Nicholas Adelman Marin Simina
Department of Electrical Engineering and Computer Science
Tulane University
New Orleans, LA 70115
e-mail: adelman, simina@eecs.tulane.edu

Abstract

In recent years, Latent Semantic Indexing (LSI) has been recognized as an effective tool for Information Retrieval in text documents. The level of "granularity" in LSI (i.e. whether LSI is performed on documents, paragraphs, sentences, phrases, etc.) is somewhat of a limiting factor, in that LSI comparisons can only be made at the level of granularity chosen. Here we argue that, as long as a record of the document structure is maintained, the level of granularity may be arbitrarily fine while still allowing for comparison at any coarser granularity. It is shown that the reduced-dimension vector for any particular section of a document is a function of the vectors of its constituent sub-sections. Using this information, we illustrate how LSI can be used to compare documents at multiple structural levels. One possible application (automated plagiarism detection) is discussed as an example of how this method of multi-level comparison may be used to improve query time in fine-granularity LSI applications.

Introduction

Latent Semantic Indexing (LSI) is a statistical corpus-based indexing technique that extends the vector-space model employed by many standard Information Retrieval (IR) techniques by incorporating a dimensional reduction step (Landauer, Foltz, and Laham 1998). As applied to textual corpuses, LSI is performed at specific granularity. "Granularity" refers to the size of the sections of text included in the corpus (i.e. documents, paragraphs, sentences, and so on). Consequently, document comparisons are limited to the granularity chosen. We are proposing a method that allows document comparisons in LSI to be performed at multiple structural levels, while only having to actually perform LSI at a single granularity. For LSI applications where fine-grain comparisons (queries) are required, this method can be used to improve the query time. One such application – an LSI-based plagiarism detection system – will be outlined to illustrate how this method can be effectively applied.

Granularity effectively determines the depth of comparison that can be performed with LSI, and thus may be viewed as a limiting factor in the utility of the application. If document granularity is used, then only coarse grain comparisons of documents can be made. If a finer granularity is used, such as sentence or phrase granularity, then only fine grain comparisons can be made. In some cases, however, it might be useful to be able to make both coarse and fine grained comparisons.

While performing LSI with coarse granularity may limit the utility of LSI, we have found that under certain conditions the granularity may be arbitrarily fine while still allowing coarse grain features of the documents to be utilized. This conclusion is reached via a slightly modified view of centroid vector clustering, which we refer to as the **sum-of-mass** method: If document trees are maintained for the contents of some corpus, then the vector for any non-leaf node in the tree can be determined exactly from the sum of its children's vectors. For instance, if LSI is performed with sentence granularity (i.e. the low-dimensional vectors for sentences are computed), the exact vectors for paragraphs can be quickly computed as a sum of their constituent sentence vectors. Essentially, this is just an exercise in controlled clustering: the document tree structure contains a specific map of known clusters within the document (paragraphs are clusters of sentences, sections are clusters of paragraphs, and so on). The implication is that documents could be compared at multiple structural levels even though LSI is only performed at one level of granularity.

The remainder of this paper is organized as follows: In the next section, a short description of the LSI method is offered. In section 3, we present a short proof of the sum-of-mass method. In section 4, an example application that utilizes the sum-of-mass method is presented and analyzed. In section 5 preliminary results are discussed, and section 6 presents some conclusions and ideas for future research.

Latent Semantic Indexing

In LSI, the projection of the corpus contents into a lower dimensional space reveals semantic *concepts* within the corpus (Landauer, Foltz, and Laham 1998). In other words, terms are abstracted to their conceptual meaning, while documents are abstracted to a weighted sum of their constituent terms' conceptual meaning. Revealing the semantic concepts within the corpus enables LSI to address the issues of synonymy, and to a lesser degree, polysemy (Papadimitriou et al. 1998). In accordance with standard IR techniques, term and document vectors in this low-dimensional space may be compared via one of a number of vector comparison techniques. Common comparison metrics include Euclidian distance and cosine similarity (Park and Elden 2003).

As applied to text documents, LSI begins with an *m*X*n* term-by-object matrix, where *m* is the number of terms and *n* is the number of objects (documents, paragraphs, sentences, etc.) in the corpus. Within this matrix, rows represent terms and columns represent objects. The values in any given column represent the term frequencies within a single object. Conversely, the values in any given row represent the term co-occurrence over all objects. The term frequencies may be weighted locally and globally to adjust the relative importance of the terms (Berry, Dumais, and O'Brien 1995). Preparing the term-by-object matrix from text documents generally requires a substantial amount of pre-processing (Landauer, Foltz, and Laham 1998).

Latent Semantic Indexing is most commonly performed using the Singular Value Decomposition (SVD). Performing the SVD on a matrix *A* decomposes it into the product of three matrices:

$$A = U\Sigma V^T \tag{1}$$

The columns of *U*, referred to as the left singular vectors, define the orthonormal eigenvectors of AA^T. Similarly, the columns of *V*, referred to as the right singular vectors, define the orthonormal eigenvectors of A^TA. *U* contains the singular vectors for the terms, while *V* contains the singular vectors for the objects. The singular values of *A*, which are the non-negative square roots of the eigenvalues of AA^T, are contained in decreasing order along the diagonal of Σ.

Once the SVD has been performed on *A*, dimensional reduction is achieved simply by setting all but the *k* largest singular values to zero (Landauer, Foltz, and Laham 1998). The reduced dimension representations of *U*, *V*, and Σ are U_k, V_k, and Σ_k, respectively.

From the matrices U_k, V_k, and Σ_k, the following matrix is defined:

$$A_k = U_k \Sigma_k V_k^T \tag{2}$$

where A_k is the best rank-*k* estimation of *A* (Landauer, Foltz, and Laham 1998).

U_k, V_k, and Σ_k define a *k* dimensional space, which will be referred to as the **LSI space** or **Semantic space**. Each term and document can be represented as a vector within this space. The vectors for the terms are obtained by scaling columns of U_k by the singular values. Similarly, the vectors for the documents are obtained by scaling the columns of V_k by the singular values (Berry, Dumais, and O'Brien 1995).

Comparisons between entities in the LSI space (both terms and objects) are most often made by calculating the cosine similarity between vectors in the space (Landauer, Foltz, and Laham 1998). A more prevalent application is to "cast" a query document into the LSI space, and use the cosine similarity measure to identify objects or terms in the space that are semantically similar to the query (Park and Elden 2003). A query object is pre-processed in the same fashion that the corpus objects were, resulting in a vector, *q*, containing the weighted term frequencies within the query. To cast the query into LSI space, the following formula is used:

$$q' = q^T U_k \Sigma_k^{-1} \tag{3}$$

For a complete discussion of the Singular Value Decomposition, readers are referred to (Golub and Van Loan 1996). For a more detailed discussion of Latent Semantic Indexing, readers are referred to (Landauer, Foltz, and Laham 1998) and (Berry, Dumais, and O'Brien 1995).

Proof of Sum-of-Mass Method

This proof begins with a term-by-object matrix *A* populated with *n* objects and *m* terms. For ease of reading, colon notation is used, rather than matrix subscript notation. For instance, the i^{th} row of matrix *A* is written *A*(*i*,:), and the i^{th} column of *A* is written *A*(:,*i*).

Assume that the *n*th object in *A* consists of the sum of the other *n*-1 objects. In other words,

$$A(:,n) = \sum_{i=1}^{n-1} A(:,i) \tag{4}$$

After the SVD has been performed, *A* is expressed as the product of three matrices, as shown in (1). Equivalently, (1) can be expressed using the *SVD expansion* (Golub and Van Loan 1996):

$$A = \sum_{i=1}^{r} \sigma_i U(i,:) V^T(i,:) \tag{5}$$

where *r* is the rank of the *A*. For any given object in *A*, the contents of the column can be expressed in terms of *U*, *V*, and Σ with:

$$A(:,j) = \sum_{i=1}^{r} \sigma_i U(i,:) V^T(i,j) \tag{6}$$

Using the assumption from (4), the following relations are derived:

$$A(:,n) = \sum_{i=1}^{r} \sigma_i U(i,:) V^T(i,n) \tag{7}$$

$$\sum_{h=1}^{n-1} A(:,h) = \sum_{i=1}^{r} \sigma_i U(i,:) \sum_{j=1}^{n-1} V^T(i,j) \tag{8}$$

From (4), (7) and (8) it is found that, in this situation,

$$V^T(i,n) = \sum_{j=1}^{n-1} V^T(i,j) \tag{9}$$

where *r* is the rank of the matrix *A*, and $1 \le i \le r$.

Equations (7), (8) and (9) generally illustrate that additive relationships that hold between objects in *A* will also hold between the corresponding right singular vectors within *V*. This implies that if object *i* in *A* is the union of a number of other objects in *A*, then the right singular vector for object *i* will be the sum of the right singular vectors for the constituent objects. For the remainder of this discussion, these constituent objects will be referred to as **sub-objects**, and their union will be referred to as the **super-object**.

Up to this point, it has been assumed that the super-object was included in *A*, and therefore played a direct role in the SVD. However, it can be seen from equation (9) that the right singular vector for the super-object can be derived from the right singular vectors of the sub-objects.

In order to see how the preceding relationship may be helpful, first consider how the vector for an object in the semantic space is calculated. The vector for a single object, d_i, in the *r* dimensional LSI space is calculated as:

$$d_i = V(i,:)\Sigma \tag{10}$$

Going back to the super-object/sub-object situation stated in (4), the vector for the super-object, d_n, can now be represented as follows:

$$d_n = (\sum_{i=1}^{n-1} V(i,:))\Sigma \tag{11}$$

Equation (11) illustrates that the exact vector of the super-object in the multi-dimensional space can be calculated from the sum of the vectors of its constituent sub-objects.

For application to LSI, it must be shown that the stated sub-object/super-object relationships hold after dimensional reduction has been performed. Equations (7) and (8) show that an original object vector in *A* can be reconstituted by summing the values of $\sigma_i U(i,:)V^T(i,:)$ over all values of *i* (i.e. over all dimensions). Dimensional reduction yields the expression in equation (2). The corresponding adjustment for equation (6) is:

$$A_k(:,j) = \sum_{i=1}^{k} \sigma_i U_k(i,:)V_k^T(i,j) \tag{12}$$

Equations (6) and (12) illustrate that the values for each object vector in *A* and A_k are comprised of the sum of number of "dimensional components". Let us refer to the d^{th} dimensional component in *A*(:,*i*) as *A(:,id)*. Similarly, A_k(:,*id*) is the d^{th} dimensional component in A_k(:,*i*). From equations (6) and (12), the following can be deduced:

$$A(:,id) = \sigma_d U(d,:)V^T(d,i) \tag{13}$$

$$A_k(:,id) = \sigma_d U(d,:)V^T(d,i) \tag{14}$$

Since equations (13) and (14) are the same, it can be concluded that dimensional reduction does not perturb the super-object/sub-object relationships. The only difference is that, after dimensional reduction, there are fewer dimensional components to sum up.

The major conclusion that can be drawn from the preceding proof is that the super-object does not need to be included in the original matrix *A* as long as all of its constituent sub-objects are.

An important issue should be addressed at this point. The components of the SVD (left singular vectors, right singular vectors, and singular values) will differ between the situation where the super-object is included in *A* and the situation where it is not. However, the relationships expressed in equations (13) and (14) will still hold, and therefore so will the sub-object/super-object relationship.

An Application

There are two categories of LSI-based applications where the sum-of-mass method may be useful. The first category includes applications where documents need to be compared at multiple levels (compare sentences to paragraphs, paragraphs to documents, and so on). The second category includes any application that requires fine-granularity LSI comparisons. This includes applications where LSI is applied at the phrase, sentence, or possibly paragraph level. This second category will be emphasized here.

Plagiarism Detection via LSI – An Overview

Past applications of LSI have most often been performed using document granularity. This has been effective for performing such tasks as document retrieval (Berry, Dumais, and O'Brien 1995) where a query phrase is presented, and semantically similar documents in the LSI space are returned. Another possible application of LSI is to compare inter-corpus documents in an attempt to determine the similarity between them. An example of such an application has been developed by (Maletic and Marcus 2000).

We now present a potential application where LSI is used to perform "fine granularity" comparison between documents. This application – a plagiarism detection system – utilizes the sum-of-mass method to speed up query time.

Plagiarism detection requires comparison of documents at a high level of detail. A number of plagiarism detection systems have been developed, many of which are discussed in (Clough 2000). One of the major obstacles in plagiarism detection is to overcome the issue of paraphrasing, and consequently the issues of synonymy and polysemy (Clough 2000). For this reason, LSI is an attractive method for detecting plagiarism.

Plagiarism may occur at many levels within a document, ranging from plagiarism of a phrase to an entire document. The most logical granularity to employ for an LSI-based plagiarism detection system would be sentence granularity. Comparing larger document structures (paragraphs, sections, or entire documents) in LSI space may not be adequate to detect plagiarism – the fact that two documents are closely related in the semantic space does not entail that one is plagiarized from the other. On the other hand, if a number of sentences (especially sentences that are temporally close) within two documents are closely related semantically, it may be more indicative of plagiarism. Furthermore, utilizing the sum-of-mass method, comparison at any coarser granularity is still possible.

A plagiarism detection system first requires a corpus of "original" documents. LSI would be performed on this corpus, thus defining the semantic space. "Suspect" documents are then presented as queries and compared to the existing corpus. If a suspect document, or some portion of it, is found to have a very close semantic relationship to portions of one or more of the original documents, then plagiarism may have occurred. Since plagiarism is a legal issue, this system will not present a definitive judgment regarding whether plagiarism actually occurred. The purpose of the system is to identify "highly suspect" document sections, and present them for further investigation.

Plagiarism Detection System Setup

During pre-processing, a tree map of each document's structure is built. For simplicity, we only consider three structural levels here – document, paragraph, and sentence. LSI is performed at sentence granularity, and the sum-of-mass method is used to derive the LSI vectors for coarser granularity document structures from the right singular vectors for the sentences. An outline of this scheme is illustrated in Figure 1.

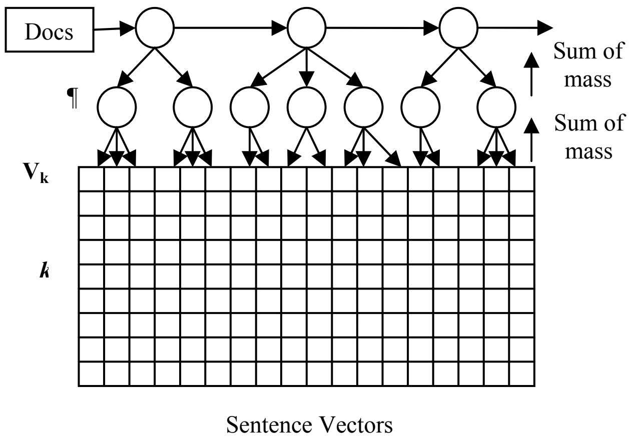

Figure 1. Corpus Setup for Plagiarism Detection

Queries are processed in the same manner. This is illustrated in Figure 2.

Querying the System

Using the document trees and sum-of-mass method, we are able to perform a simple greedy search on the corpus in order to identify possible plagiarisms. The search begins by querying at document-level granularity. The derived vector for the query document is compared to each document vector in the corpus, using the cosine-similarity metric. If the similarity between the query and any document in the corpus falls within a specified threshold, then the corpus document is selected for further examination.

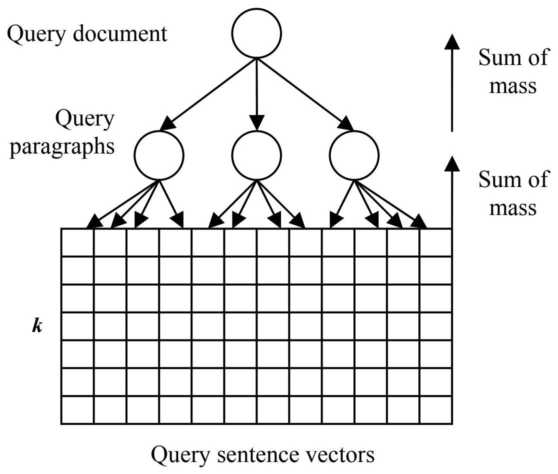

Figure 2. Query Setup for Plagiarism Detection

Next, the selected corpus documents and the query document are compared at paragraph-level granularity. Each query paragraph is compared to each paragraph within the selected corpus documents. Once again, if the similarity between a query paragraph and any paragraph within the corpus falls within a given threshold, then the corpus paragraph is selected for further investigation.

The final step is to compare at sentence-level granularity. For each query paragraph, q_{pi}, its constituent sentences are only compared to the sentences within corpus paragraphs that q_{pi} was highly similar to. A link between highly similar sentences is created to facilitate further investigation.

An outline of the entire query procedure is illustrated in Figure 3. The simple greedy search approach that is employed can be viewed as a series of filters that allow unnecessary vector comparisons to be avoided. For instance, the document level comparison filters out semantically dissimilar documents, so further comparison are only performed on "documents of interest". Considering that the cosine similarity is an O(k) operation, avoiding unnecessary comparison will greatly improve the query time.

Examining an Alternative

An alternative to using the sum-of-mass method is to rely on **folding-in** documents into the Latent Semantic space (Berry, Dumais, and O'Brien 1995). This method will be referred to as the **folding method**. Folding-in a document is the same as casting a query vector into a pre-defined LSI space. The idea is to perform LSI at document granularity on the corpus, and then fold-in each document's constituent paragraphs and sentences to derive their respective vectors in the LSI space.

Query documents would be treated similarly. The query document, each constituent query paragraph, and each constituent query sentence would be cast into the LSI space before the query could proceed. Once the query pre-processing is complete, the query algorithm would be exactly the same as the algorithm in Figure 3.

```
Greedy_Multi_Level_Compare( $d_q$ )
    for each document, $d_i$, in the corpus
        if cosine-similarity( $d_q$ , $d_i$ ) > $\delta$
            for each paragraph, $p_{qj} \in d_q$
                for each paragraph $p_i \in d_i$
                    if cosine-similarity( $p_{qi}$ , $p_i$ ) > $\delta$
                        for each sentence $s_{qi} \in p_{qi}$
                            for each sentence $s_i \in p_i$
                                if cosine-similarity( $s_{qi}$ , $s_i$ ) > $\delta$
                                    create-link( $s_{qi}$, $s_i$ )
$d_i$ = $i^{th}$ document in corpus                  $d_q$ = query document
$p_i$ = $i^{th}$ paragraph in a corpus document      $p_{qi}$ = $i^{th}$ paragraph in $d_q$
$s_i$ = $i^{th}$ sentence is a corpus paragraph      $s_{qi}$ = $i^{th}$ sentences in a query paragraph
$\delta$ = threshold
```

Figure 3. Simple Greedy Search Algorithm For Multi-Level Comparison

The main advantage of the folding method lies in the fact that the SVD would be calculated for a much smaller corpus matrix (obviously a term-by-document matrix will be smaller than a term-by-sentence matrix). The SVD is a very computationally intensive calculation, so in this respect, the folding method should enjoy a great advantage

Although the setup time for the corpus should be improved, this is the only clear advantage that the folding method holds over the sum-of-mass method. It should be noted that the corpus setup only needs to be performed once, but the LSI space may be queried innumerable times. Therefore, a computational advantage in query time should be more valuable than an advantage in corpus setup time.

Both the sum-of-mass method and the folding method use the algorithm in Figure 3 to perform queries, so they are computationally equivalent in this respect Therefore, the only area where an advantage may be found is in the query setup time. Here we find that the sum-of-mass method holds a distinct computational advantage over the folding method.

First note that folding-in is an O(mk^2) operation, while performing the sum-of-mass is O(k). In both methods the query setup requires that the query sentences be cast into the LSI space. However, for the folding method, the query paragraphs and query document must also be folded in. Conversely, for the sum-of-mass method, the LSI vectors for the query paragraphs and query document are computed via the sum-of-mass. Consequently, the sum-of-mass method enjoys a computational advantage of an order of magnitude in m and k. Over many queries this will be a great computational advantage.

In addition to the computational advantage of the sum-of-mass method, there is also a theoretical advantage. It has been noted in (Landauer, Foltz, and Laham 1998) and (Berry, Dumais, and O'Brien 1995) that folding-in documents perturbs the orthogonality of the right singular vectors, slightly distorting the semantic space. Thus, the position of the LSI vector for the folded document, relative to the positions of the vectors for the original corpus documents, will deviate slightly from where it should be. In other words, folding-in a document yields an **estimate** of the document's position in the semantic space. On the other hand, the proof of the sum-of-mass method indicated that this method could be used to determine the **exact** position of a super-document in the semantic space. Therefore, the sum-of-mass method could be viewed as more accurate than the folding method.

Preliminary Experimental Results

We have performed one preliminary experiment designed to test the sum-of-mass method and compare it to the folding method. The corpus used in these preliminary tests consists of 15 essays related to World War Two. These essays were taken from a "paper mill" website that offers essays for download (http://www.planetpapers.com). During pre-processing, global weighting, local weighting, and normalization were not performed.

LSI was performed on the corpus using both methods, and the resulting semantic spaces were queried using the same query documents. Queries proceeded via the algorithm outlined in figure 3, with one modification. Rather than using a single threshold, d, at each level of comparison, three different thresholds were used. These thresholds ranged from relatively low (0.75) for document level comparisons to high (0.95) for sentence-level comparisons.

		Sum-of-mass		**Folding**	
Query	**Level**	**Recall**	**Precis**	**Recall**	**Precis**
0	Doc	1.0	1.0	1.0	1.0
	Para	1.0	0.65	1.0	0.14
	Sent	0.98	0.84	1.0	0.26
1	Doc	1.0	1.0	1.0	1.0
	Para	1.0	0.71	1.0	0.44
	Sent	1.0	1.0	1.0	0.74
2	Doc	1.0	1.0	0.5	1.0
	Para	1.0	0.68	0.5	0.07
	Sent	0.99	0.91	0.57	0.15

Table 1. Experimental results for the two methods

The first two queries presented were exact copies of two of the papers in the original corpus. The third query was a union of the first two queries. The recall and precision for each query at each structural level is presented in Table 1.

Evaluation and Discussion

The results of the preliminary experiment reveal two things. First, the difference in recall between the methods is almost negligible (the low recall for the folding method in query 2 is due to the fact that one of the two relevant documents was not identified during the greedy search). Second, there are obvious differences in the precision of the two methods. This phenomenon may be attributable to a number of factors.

One possible factor is the "correctness" of the sum-of-mass method versus the folding method, mentioned on the previous page.

A second possible factor is the granularity at which LSI was performed. It seems that performing LSI at sentence granularity may expose specific term relationships, while performing LSI at document granularity may expose more general term relationships. If this is the case, it would explain the differences in precision between the methods.

A final factor is the size of the corpus. With such a small corpus size, it is difficult to determine how reliable these results are. More experimentation with a larger corpus is needed.

Aside from experimentation with a larger corpus, a number of other experiments need to be performed in order to further evaluate our proposed methods. First of all, the subjects of all of the documents in our preliminary experiment were very similar. Testing with a more diverse corpus should also be performed. Secondly, our preliminary tests evaluated whether the system would accurately identify blatant plagiarisms (exact copies). Further tests should be engineered to see if the system can detect paraphrasing and other subtle forms of plagiarism.

Conclusions

We have shown that using the sum-of-mass method to augment the standard LSI method with structural information may allow the capabilities of Latent Semantic Indexing to be extended. However, a number of issues still need to be addressed and/or resolved.

One issue that we have not thoroughly investigated is how local weighting, global weighting, and normalization on the term-by-document matrix will affect the sum-of-mass method. Methods for local and global weighting are discussed in (Landauer, Foltz, and Laham 1998) and (Berry, Dumais, and O'Brien 1995). The main issue here is whether weighting and normalization will perturb the super-object/sub-object relationship. Intuitively speaking, global weighting should not affect this relationship, as weights are applied uniformly across the objects and terms. Normalization, on the other hand, would almost certainly alter the super-object/sub-object relationship: after normalization, the union of the sub-objects would no longer be equal to the super-object. This is an area where further investigation is needed. Local weighting at the sentence level would also perturb the super-object/sub-object relationship. However, local weighting could be performed at the document level, and the weights could be propagated down to the sentence level.

Another issue to investigate is how large a term-by-sentence matrix may be before performing the SVD becomes computationally infeasible. The number of terms used in LSI may be as great as 100,000, and the number of documents may be just as large. If a corpus contains 100,000 documents, each containing an average of 50 sentences, then we would be looking at performing the SVD on a 100,000x5,000,000 matrix, which may be computationally infeasible.

Perhaps one of the most promising applications of the sum-of-mass method relates to the fact that it allows structural information to be easily combined with semantic information. It has been noted by many authors that a drawback to LSI is that it is inherently non-order-preserving (Landauer, Foltz, and Laham 1998). The sum-of-mass method preserves structural information that could be used in conjunction with LSI to simultaneously perform semantic and structural comparisons between objects. Possible applications might include DNA or Software analysis using LSI.

References

Landauer, T. K., Foltz, P. W., and Laham, P. 1998. Introduction to Latent Semantic Analysis. *Discourse Processes* 25:259-284.

Papadimitriou, C., Raghavan, P., Tamaki, H., and Vempala, S. 1998. Latent Semantic Indexing: A Probabilistic Analysis. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 159-168. Seattle, Washington:ACM Press.

Park, H., and Eldén, L. 2003. Matrix Rank Reduction for Data Analysis and Feature Extraction. Technical Report, 03-015, Dept. of Computer Science, University of Minnesota-Twin Cities.

Berry, M. W., Dumais, S. T., and O'Brien, G. W. 1995. Using Linear Algebra for Intelligent Information Retrieval. *SIAM: Review* 37(4):573-595.

Golub, G. H., and Van Loan, C. F. 1996. *Matrix Computations, Third Edition*. Baltimore, Maryland: The Johns Hopkins University Press.

Maletic, J. I., and Marcus, A. 2000. Using Latent Semantic Analysis To Identify Similarities in Source Code to Support Program Understanding. In *Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence*, 46-53. Vancouver, Canada: IEEE Press.

Clough, P. 2000. Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies. Internal Report CS-00-05, Dept. of Computer Science, University of Sheffield.