

Latent Semantic Analysis and Fiedler Retrieval*

Bruce Hendrickson[†]

September 21, 2006

Abstract

Latent semantic analysis (LSA) is a method for information retrieval and processing which is based upon the singular value decomposition. It has a geometric interpretation in which objects (e.g. documents and keywords) are placed in a low-dimensional geometric space. In this paper, we derive an alternative algebraic/geometric method for placing objects in space to facilitate information analysis. We show that our method is closely related to LSA, and essentially equivalent for particular choices of scaling parameters. We then show that our approach supports a number of generalizations and extensions that existing LSA approaches cannot handle.

1 Introduction

Latent semantic analysis (LSA) is a well-known tool for information retrieval and analysis. The canonical example of LSA begins with a term–document matrix in which matrix rows correspond to key-words or terms, and matrix columns are documents. A nonzero value in the matrix means that the corresponding document contains the corresponding term. This vector space model of information is due to Salton [18].

Instead of working directly with this matrix, LSA replaces it with a low rank approximation using the singular value decomposition [9]. A variety of interpretations of LSA have been proposed. It is a noise reduction technique in which only the most significant parts of the term–document matrix are retained. Alternatively, it is a method for mapping terms and documents into geometric spaces, after which geometric algorithms can facilitate analysis.

The purpose of this paper is to derive a novel algebraic algorithm for placing terms and documents in space, which we call *Fiedler retrieval* for reasons that will become clear. Whereas traditional LSA is motivated by a matrix approximation argument, our alternative follows from a geometric optimization problem. We show that Fiedler retrieval is algebraically very closely related to LSA. Besides providing a fresh perspective on LSA, we show that our approach allows for novel generalizations and extensions that are not possible with traditional approaches. For instance, our methodology supports queries that involve both terms and documents, e.g. “return documents with these terms **and** similar to these documents”. As another example, unlike existing LSA techniques, our approach allows

*This work was supported by the LDRD program at Sandia National Laboratories. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE under contract number DE-AC-94AL85000.

[†]Discrete Algorithms & Math Dept., Sandia National Labs. Email: bah@sandia.gov

for the consideration of term–term and document–document similarities in generating the geometric embedding. The former might come from a thesaurus or multilingual dictionary, while the latter could be provided by co-citation or link analysis.

In §2 we review the intuition and mathematics underlying traditional LSA. In §3 we derive Fiedler retrieval by motivating and then solving a simple geometric optimization problem. In §4 we discuss the algebraic relationship between LSA and Fiedler retrieval. Finally, in §5, we discuss some of the benefits of our alternative derivation, including some ideas for extending LSA in new ways.

As we discuss in §3, the geometric problem we use to motivate our approach reduces to the calculation of eigenvectors of the Laplacian matrix of a graph. Laplacian eigenvectors have been used in a wide range of applications in combinatorial optimization including graph partitioning [13], clustering [10], and linear arrangement [14]. In many applications, only the eigenvector corresponding to the smallest non-zero eigenvalue is of interest. But several applications have involved the use of multiple eigenvectors [1, 3, 10], and our approach will too. For some of the development in this paper, we will be interested in the Laplacian eigenvectors of bipartite graphs. The resulting matrix has a natural 2×2 block structure. Others have used Laplacian eigenvectors of bipartite graphs e.g. Berry, et al. for reordering term/document matrices [8], Dhillon [10] and Zha, et al. [21] for data clustering, and Newton, et al. for graph drawing [17]. But to our knowledge, the connection to LSA described in this paper is new. In a paper similar in spirit to this one, Bartell, et al. have shown that LSA is equivalent to a special case of a different geometric optimization problem known as multidimensional scaling [2].

2 Latent Semantic Analysis

In this section we briefly sketch the fundamental operations in latent semantic analysis (also known as latent semantic indexing). We follow the traditional derivation in which LSA is motivated by an optimal matrix approximation. More comprehensive presentations can be found in some of the citations e.g. [9, 6, 4, 5].

The canonical example of LSA begins with a $t \times d$ term–document matrix A . Each row of A is associated with a keyword or term, and each column is a document. A matrix entry $A(i, j)$ is a non-negative value which encodes the importance of the term i in the document j . A toy example of a term–document matrix is depicted in Fig. 1. There is an extensive literature on methods for generating such a matrix from a corpus of documents, but the construction process is beyond the scope of the current paper. Often, the term–document matrix is scaled to achieve some attractive normalization property, or to weight some terms or documents more heavily than others. Thus, we will consider the more general *scaled* term–document matrix $B = D_t A D_d$, where D_t and D_d are non-negative diagonal matrices of size $d \times d$ and $t \times t$ respectively.

In this *vector space* model of information, a document is described as a (weighted) vector of terms of which it is comprised. Two documents are similar if the inner product of their vectors is large. Thus, the matrix $B^T B$ describes the set of inter-document similarities. A query q is also a (weighted) vector of terms. The answer to a query is a set of documents that are similar to it, e.g. documents whose vectors have large inner-products with the query vector, that is large values in $B^T q$. (Most commonly, similarity is measured in angular distance, i.e. as a direction cosine between the two vectors, which is just a normalized

$$\begin{array}{c} \text{Documents} \\ \text{Terms} \end{array} \begin{pmatrix} 2 & & 4 \\ 3 & 3 & \\ & 4 & 2 & 1 \end{pmatrix}$$

Figure 1: Example of a sparse, 3×4 term–document matrix.

version of the inner product.)

Unfortunately, this simple model has a variety of well-known shortcomings. Most notably, small differences in vocabulary (e.g. *car* instead of *automobile*) can make documents look different from queries, even if their topics are overlapping. LSA attempts to address this problem through compression and noise reduction. Specifically, LSA uses matrix transformations to retain only the most significant portions of B , and then performs queries in this transformed space.

More formally, LSA is constructed around the singular value decomposition (SVD) of B ,

$$B = U\Sigma V^T,$$

where U and V are orthogonal matrices and Σ is diagonal and non-negative. The diagonal values of Σ are ordered to be non-increasing. In LSA, the matrix is approximated by a truncated SVD in which the first k diagonal values of Σ are retained, but the rest are set to zero. That is,

$$B \approx B_k = U_k \Sigma_k V_k^T,$$

where U_k is $t \times k$, Σ_k is $k \times k$ and V_k is $d \times k$. The truncated SVD is the best rank k approximation to B in the Frobenius norm.

The truncated SVD can be thought of as generating a k -dimensional embedding of the terms and documents. However, it is important to note that the term coordinates and the document coordinates are distinct entities. The notation in the field is inconsistent with respect to the scaling factors, but we choose to define the columns of $\Sigma_k^{1/2} V_k^T$ as the *document vectors* and $\Sigma_k^{1/2} U_k^T$ as the *term vectors*.

With the truncated SVD approximation to B , the inner-products required for document–document similarities can now be approximated as $B^T B \approx B_k^T B_k = (\Sigma_k^{1/2} V_k^T)^T \Sigma_k (\Sigma_k^{1/2} V_k^T)$, that is, as the inner product of document vectors, scaled by Σ_k .

Given a query vector q , we want to embed q into the document space in such a way that inner products with document vectors approximate $B^T q$. It is straightforward to see that this is achieved by letting the transformed query vector \hat{q} be $\Sigma_k^{1/2} U_k^T q$, with a standard, unscaled inner product. (N.B. Alternatively, we could have used an inner product scaled by Σ_k as for document–document comparisons, in which case \hat{q} would be $\Sigma_k^{-1/2} U_k^T q$. Both approaches lead to interpretation challenges since either the scaling of queries differs from that of documents, or the inner product differs for different kinds of questions.)

These procedures are summarized in Figure 2.

LSA document embedding:**Given** scaled term–document matrix B .

- (1) Compute truncated SVD of B , $U_k \Sigma_k V_k^T$.
- (2) Assign the position of document i to be $\Sigma_k^{1/2} V_k^T e_i$.

Querying:**Given** document embedding, and query vector q .

- (1) Compute query location = $\Sigma_k^{1/2} U_k^T q$.
 - (2) Return documents nearest to query point (nearest in angular distance).
-

Figure 2: LSA algorithms for term/document embeddings and querying.

3 Graphs, Laplacian Eigenvectors, and Fiedler Retrieval

As sketched above in §2, the traditional derivation of LSA is based upon optimal matrix approximations. But informally, the method succeeds in mapping documents into geometric space in such a way that similar documents are close to each other. In this section, we pick up on this geometric closeness objective and develop an alternative algebraic method that explicitly tries to optimize closeness. For reasons that will be clear at the end of this section, we will call our approach *Fiedler retrieval*. In §4, we discuss the mathematical relationship between traditional LSA and our methodology.

Our approach begins with a graph $G = (V, E)$ in which V is a set of vertices and E is a set of vertex pairs known as edges. An edge (i, j) connecting vertices i and j has a non-negative weight $w_{i,j}$ which describes how similar the two vertices are. Larger weights correspond to a greater degree of similarity or affinity. The vertices may represent several different classes of objects. For instance, in §4 we will look at the special case where the vertices are terms and documents, and the similarities are entries of the scaled term–document matrix B . But for now, we will consider the more abstract and general problem. We will assume that the graph is *connected* — that for any two vertices there is a path connecting them.

Our goal is to place the vertices of the graph into a low-dimensional geometric space in such a way that similar vertices are close to each other (i.e., edge lengths will be short). Geometric embeddings of graphs can be useful for a variety of reasons, but for the purposes of this paper, our eventual goal is the same of the goals of LSA. We hope to use geometric proximity as a way to identify vertices that are similar to each other, even if they don't have an edge between them.

The geometric embedding problem can be posed as an algebraic minimization. There are many ways to mathematically describe such an embedding, but one will be particularly useful. Specifically, we choose to find points in a k -dimensional space that minimize the weighted sum of the square of edge lengths. That is, if p_r is the location of vertex r , then

$$\text{Minimize } \sum_{(r,s) \in E} w_{r,s} |p_r - p_s|^2.$$

If the number of vertices is n , and the geometric space has dimensionality k , then the positions of the vertices can be considered to be an $n \times k$ matrix X . Define the *Laplacian*

matrix L as follows.

$$L(i, j) = \begin{cases} -w_{i,j} & \text{if } e_{ij} \in E \\ \sum_k w_{i,k} & \text{if } i = j \\ 0 & \text{Otherwise.} \end{cases}$$

That is, the Laplacian is the negative of the matrix of weights, except that the diagonal values are chosen to make row-sums zero. Note that L is symmetric and positive semi-definite. After a bit of algebra, our minimization problem can be rewritten as

$$X = \operatorname{argmin} \operatorname{Trace}(X^T L X). \quad (1)$$

This minimization problem is poorly posed for three reasons. First, it is invariant under translations. To avoid this problem we can add a constraint to make the median of the point set be the origin. For generality, we will allow different vertices to be weighted differently. For instance, a vertex representing a document might be weighted differently from a vertex representing a term. We allow this flexibility by including a positive diagonal weighting matrix D in our normalization. That is,

$$\text{(Constraint 1) for } i = 1, \dots, k \quad X_i^T D I^n = 0 \quad (2)$$

where I^n denotes the vector of n ones.

Second, even with this constraint the minimization problem has the trivial solution of placing all the vertices at the origin. To avoid this, we can simply insist that the weighted sum of squares of each coordinate value is nonzero. That is,

$$\text{(Constraint 2) for } i = 1, \dots, k \quad X_i^T D X_i = \delta_i \quad (3)$$

for some positive values δ_i . Without loss of generality, we will choose to order the axes so that the δ_i values are non-increasing. We will have more to say about these values when we compare Fiedler retrieval to LSA in §4.

Finally, we want to ensure that each coordinate conveys distinct information. We accomplish this by imposing the constraint that the vector of coordinate values in each dimension is orthogonal to the coordinate values from any other dimension. Again, we allow for different vertices to be weighted differently.

$$\text{(Constraint 3) for } i \neq j \quad X_i^T D X_j = 0. \quad (4)$$

Denoting the diagonal matrix of δ values by Δ , we can combine constraints 2 and 3, resulting in the following optimization problem.

$$X = \operatorname{argmin} \operatorname{Trace}(X^T L X) \quad (5)$$

Subject to :

$$(i) \quad X_i^T D I^n = 0 \text{ for } i = 1 \dots k,$$

$$(ii) \quad X^T D X = \Delta$$

Consider the generalized eigenproblem $Ly = \lambda Dy$. L is positive semi-definite, and it has a generalized eigenvector proportional to I^n with eigenvalue 0. If, as we assume, the graph is connected, then all other generalized eigenvalues are positive [11, 12]. Sort these generalized eigenvalues λ_i in non-decreasing order, and form the matrix $\Lambda = \operatorname{diag}(\lambda_i)$. Order the corresponding eigenvectors q_i in the same way and combine them to form a matrix Q . It follows from elementary properties of the generalized eigenproblem that $L = DQ\Lambda Q^T D$, and $Q^T D Q = I$. Let \tilde{Q}_k denote the matrix $[q_2, \dots, q_{k+1}]$.

Theorem 3.1 A solution to the minimization problem (5) is $X = \tilde{Q}_k \Delta^{1/2}$.

Proof:

Let $X = QZ$ for some $n \times k$ matrix Z . The constrained minimization problem in 5 can now be rewritten as

$$Z = \operatorname{argmin} \operatorname{Trace}(Z^T Q^T L Q Z) = \operatorname{Trace}(Z^T \Lambda Z) \quad (6)$$

Subject to :

$$(i) Z^T Q^T D 1^n = 0$$

$$(ii) Z^T Z = \Delta$$

Since $q_1 = 1^n$, constraint (i) of Eq. (6) can be simplified to become $0 = Z^T Q^T D q_1 = Z^T e_1$. Thus, the first row of Z is zero.

A solution to (6) should involve only the smallest of the eigenvalues, which are those in the leading diagonal entries of Λ . Since $Z^T e_1 = 0$, a solution can be found in the span of the unit vectors e_2, \dots, e_{k+1} .

To minimize the trace, we now need to have large values of δ_i be paired with small values of λ_i . Recall that the λ_i are non-decreasing, while the δ_i are non-increasing. Thus, the trace is minimized by having the nonzero portion of Z be the identity matrix. That is, $Z = [e_2, \dots, e_{k+1}]$. The theorem follows. ■

Note that if Δ or Λ have repeated values, then the solution to (5) is degenerate and any basis for the subspace spanned by eigenvectors corresponding to repeated values is also a minimizer.

Theorem 3.1 says that a solution to the geometric optimization problem is found when the coordinates of vertex i are the i th entries of generalized eigenvectors q_2, \dots, q_{k+1} of $Ly = \lambda Dy$, scaled by the square roots of the normalization values δ . We will call this solution a *Fiedler embedding* in honor of Miroslav Fiedler's pioneering work exploring the relationship between graphs and Laplacian eigenvectors.

3.1 Queries

Once we have embedded the graph in space, we can use geometry to identify pairs of similar vertices. Two vertices might not have an edge between them, but they might still be similar if they have many neighbors in common. Since the Fiedler embedding tries to keep edge lengths short, vertices sharing neighbors should be placed close to each other. So given a vertex, its geometrically nearest neighbors are natural candidates for similarity. In this discussion, Euclidean distance is the most natural metric, but as noted above, LSA traditionally uses angular distance, measured by direction cosines.

Now suppose we want to add a new vertex v to the geometric space, and v is known to be similar to some of the vertices we have already placed. In the language of LSA, this new vertex corresponds to a query and its known similarity values comprise a query vector. Once v is given coordinates, we could use geometric algorithms to find nearby vertices, and these would be the output of the query.

As with the derivation of the Fiedler embedding, we wish to place v into the space in such a way that it is near to vertices it is known to be similar to. Mimicking the development

above, we will position the new vertex to minimize the (weighted) sum of squares of distances to the vertices it has an edge to. That is, we wish to find a position p_v which solves

$$p_v = \operatorname{argmin} \sum_{(s,v) \in E} w_{s,v} |p_s - p_v|^2.$$

Setting the derivative to zero, it is easy to see that the solution to this problem is

$$p_v = \sum_{(s,v) \in E} w_{s,v} p_s / \sum_{(s,v) \in E} w_{s,v} = \Delta^{1/2} \tilde{Q}_k^T q / \|q\|_1.$$

where q is the vector of similarity values for the new vertex (values of w in the derivation above).

To summarize, the Fiedler retrieval processes for constructing coordinates and querying are sketched in Figure 3.

Fiedler Embedding:

Given W a set of non-negative affinities for vertex pairs,

Δ a diagonal, non-negative matrix of scalings for each coordinate,

and D a diagonal, non-negative matrix of normalization values for each vertex.

- (1) Generate Laplacian matrix L from W .
- (2) Compute $k + 1$ generalized eigenvectors q_j corresponding to the smallest generalized eigenvalues of $Lx = \lambda Dx$.
- (3) Let $\tilde{Q}_k = [q_2, \dots, q_{k+1}]$.
- (4) Assign the position of vertex i to be $\Delta^{1/2} \tilde{Q}_k^T e_i$.

Querying:

Given Fiedler embedding, and new entity with affinity/query vector q .

- (1) Compute query location $= \Delta^{1/2} \tilde{Q}_k^T q / \|q\|_1$.
 - (2) Return vertices nearest to query point (nearest in Euclidean distance).
-

Figure 3: Fiedler retrieval algorithm for graph embedding and querying.

4 Relationship Between LSA and Fiedler Retrieval

Our Fiedler retrieval algorithm was derived as the solution to a geometric minimization problem, and involves generalized eigenvectors of a Laplacian system. Latent Semantic Analysis is the solution to an optimal matrix approximation problem, and revolves around singular values and vectors. Despite these differences, a comparison of Figures (2) and (3), suggests a high degree of structural similarity between these two methods. In this section, we show that they are actually quite closely related. In particular, we show that for certain choices of the free parameters, the algebraic/geometric spaces employed by the two methods are related to each other by simple scalings.

Figures (2) and (3) reveal three distinct differences between LSA and Fiedler retrieval. First, in LSA the embedding and query operations involve scaling by the square root of the

singular values, while in Fiedler retrieval they are scaled by the square root of the δ_i values. As the δ_i values were free parameters in Fiedler retrieval, they could be chosen to match the singular values from LSA, so this difference is insignificant.

Second, the location of a query point in Fiedler retrieval involves scaling by the 1–norm of the query vector, but no such scaling appears in LSA. Normalization doesn’t matter in LSA since distances are measured in terms of direction cosines. But the normalization does have an impact in Fiedler retrieval since it uses Euclidean distances.

Third, and most important, the methods use seemingly quite different algebraic spaces. LSA uses singular vectors corresponding to large singular values of the scaled term–document matrix. But Fiedler retrieval uses generalized eigenvectors corresponding to small eigenvalues of the Laplacian matrix. In the remainder of this section, we show that these spaces are actually quite closely related.

LSA is concerned with terms and documents, while our discussion of Fiedler retrieval discussed general entities and similarities. Consider applying Fiedler retrieval to a set of documents and terms. Terms will have an affinity for the documents that contain them, and vice versa. That is, a scaled term–document matrix can be thought of as encoding the similarity values between term and document entities. With this interpretation, the term–document matrix from Fig. 1 can be described as a weighted graph as depicted in Fig. 4. Note that this graph has a *bipartite* structure; that is, no edges connect terms to terms or documents to documents.

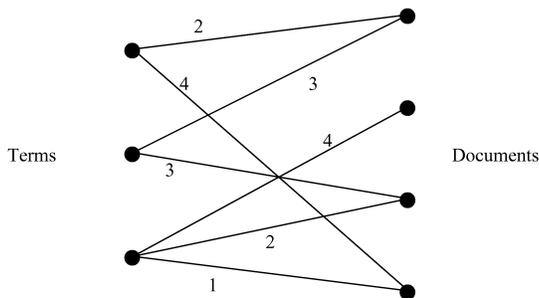


Figure 4: Graph corresponding to the term–document matrix from Fig. 1.

The Laplacian matrix corresponding to this bipartite graph is shown in Fig. 5. Note that it is square, with a row and column for each term and document. Note further that the diagonal blocks have no off–diagonal nonzeros since there are no document–document or term–term edges in the graph. The horizontal and vertical lines in the figure demarcate this block structure. The off-diagonal blocks are just the negative of the scaled term–document matrix B . That is,

$$L = \begin{pmatrix} D_1 & -B \\ -B^T & D_2 \end{pmatrix},$$

where D_1 and D_2 are diagonal matrices which make the row sums zero.

Fiedler retrieval makes use of generalized eigenvectors corresponding to small eigenvalues of the system $Lx = \lambda Dx$, where D was unspecified. To compare with LSA, we now choose to make D equal to the diagonal of L . The generalized eigenproblem underlying Fiedler

	Terms	Documents
Terms	6	-2
	6	-3
	7	-4
	-2	-3
	-3	-4
	-4	-2
	-3	-1
	-2	5
	-4	4
	-1	5
Documents	5	5

Figure 5: Laplacian matrix corresponding to the term–document matrix from Fig. 1.

retrieval can now be written as follows.

$$\begin{bmatrix} D_1 & -B \\ -B^T & D_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

With a bit of simple algebra mirroring the derivation in [10], this can be rewritten as

$$\begin{aligned} Bx_2 &= (1 - \lambda)D_1x_1 \\ B^Tx_1 &= (1 - \lambda)D_2x_2 \end{aligned}$$

Substituting $y_1 = D_1^{1/2}x_1$ and $y_2 = D_2^{1/2}x_2$, we find that

$$\begin{aligned} D_1^{-1/2}BD_2^{-1/2}y_2 &= (1 - \lambda)y_1 \\ D_2^{-1/2}B^TD_1^{-1/2}y_1 &= (1 - \lambda)y_2 \end{aligned}$$

These equations define the singular value decomposition of $D_1^{-1/2}BD_2^{-1/2}$. That is, y_1 and y_2 are the left and right singular vectors respectively with singular value $(1 - \lambda)$. More formally, we have the following result.

Theorem 4.1 *Let Laplacian be a Laplacian matrix*

$$L = \begin{pmatrix} D_1 & -B \\ -B^T & D_2 \end{pmatrix}, \quad (7)$$

where D_1 and D_2 are diagonal and let D be the diagonal of L . If $(u, v, 1 - \lambda)$ is a singular triplet of $D_1^{-1/2}BD_2^{-1/2}$, then $(D_1^{-1/2}u : D_2^{-1/2}v, \lambda)$ is a singular pair of the generalized eigensystem $Lx = \lambda Dx$, where $a : b$ denotes concatenation of a and b .

This theorem indicates that there is a close relationship between the eigenvectors in Fiedler retrieval and the singular vectors of LSA. Specifically, a scaling of the eigenvectors is equivalent to performing LSA on a scaled matrix. Note that the singular values are $(1 - \lambda)$ while the eigenvalues are λ . Thus, large singular vectors correspond to small eigenvectors. So the spaces used by LSA and Fiedler retrieval are closely connected.

If D_t and D_d , the scaling matrices from §2, are chosen so that B has constant row sums and constant column sums (i.e. Sinkhorn scaling), then D_1 and D_2 become multiples of identity matrices. In this case, the relationship between the spaces used by the two approaches is even simpler.

Corollary 4.2 *Let B from Theorem 4.1 have constant row sums and constant column sums so that $D_1 = \alpha^2 I$ and $D_2 = \beta^2 I$. Then if $(u, v, 1 - \lambda)$ is a singular triplet of $\text{diag}(\alpha^{-1})B \text{diag}(\beta^{-1})$, then $(\alpha^{-1}u : \beta^{-1}v, \lambda)$ is a singular pair of the generalized eigen-system $Lx = \lambda Dx$.*

The computation time for the two approaches is also very similar. For large, highly unstructured matrices, iterative algorithms are likely to be the methods of choice for singular and eigenvector computations. Iterative algorithms for computing the SVD, like those in Berry’s SVDPACK [7], require a multiplication by both B and B^T in each iteration. An iterative method, like those in ARPACK [16] for the larger eigenproblem in Fiedler retrieval will also involve these two matrix products. If the diagonal blocks are diagonal, then the additional work per iteration is linear in the problem size. Effective preconditioning will be important for both approaches, and the Fiedler retrieval approach may be able to benefit from sophisticated methods for preconditioning Laplacians like algebraic multigrid [20] and support theory [19].

5 Advantages of Fiedler Retrieval

As discussed in §4, Fiedler retrieval can be viewed as a close cousin to traditional LSA. We believe it provides several distinct advantages over traditional LSA formulations. It provides a simple and intuitive way to explain the power of LSA. But it also has some more concrete advantages.

- The Fiedler embedding is an explicit attempt to place similar objects near to each other. This objective is often alluded to in the LSA literature, but with traditional presentations of LSA the mathematical underpinnings of nearness are opaque.
- Unlike traditional LSA, in Fiedler retrieval terms and documents are treated equivalently and co-located in the same space. This refutes a claim made by Deerwester, et al. in the foundational LSA paper [9]. “... it is not possible to make a single configuration of points in space that will allow both between [term/document] and within [term/term] comparisons.” As itemized below, this unification creates opportunities for several extensions to traditional LSA.
 - Fiedler retrieval supports queries that include both term *and* document similarities. For example, one could search for documents similar to a few particular documents and a few specific terms. These kinds of cross-queries are problematic for traditional LSA.
 - In the standard development of LSA it is assumed that the only usable information is term/document connections. LSA does not naturally allow for inclusion of any additional information related to term/term or document/document similarities. However, nothing in the derivation of the Fiedler retrieval algorithm from §3 exploited the bipartite nature of the graph. That is, the diagonal blocks of the Laplacian matrix need not be diagonal matrices. We could have explicitly added information about document/document or term/term similarities into our construction of the geometric embedding. For instance, citation analysis

or hyperlinks could have provided document/document similarities and a thesaurus or (multilingual) dictionary could have offered term/term similarities. In a web setting, link and text analysis can be combined into a common algebraic framework. In a recommender system, product/product similarities could be included, enhancing current methods that just include consumer/product information. Fiedler retrieval allows for a principled inclusion of such information.

- LSA is traditionally constrained to capturing the relationships between two classes of objects, e.g. terms and documents. It is unclear how to extend the standard methodology to more object classes, e.g. terms, documents and authors. Fiedler retrieval is not limited in this way. The Laplacian matrix will have a logical block structure, with as many block-rows and block-columns as object classes. That is, the rows and columns of the matrix will have entries for terms, documents and authors. The diagonal blocks will capture similarities between objects of the same type (e.g. authors with authors), while the off-diagonal blocks will encode similarities between disparate types. Several researchers have recently addressed the challenge of multiple classes of objects by extending the term–document matrix to higher dimension and using multilinear algebra techniques (e.g. the TOPHITS approach of Kolda and Bader [15]). Although these approaches are mathematically elegant, computations on tensors are much more challenging than those on matrices. With Fiedler retrieval, we obtain many of the advantages of tensors, while retaining the algorithmic and mathematical benefits of working with two-dimensional matrices.

Acknowledgements

The ideas in this paper have benefitted from discussions with Tammy Kolda, Liz Jessup, Mike Berry, Inderjit Dhillon, Brett Bader, Chris Ding, Petros Drineas, and especially Erik Boman. I am particularly indebted to Inderjit for identifying an error in an earlier draft.

References

- [1] C. J. ALPERT AND S.-Z. YAO, *Spectral partitioning: the more eigenvectors, the better*, in Proc. Annual ACM Design Automation Conf., 1995, pp. 195–200.
- [2] B. T. BARTELL, G. W. COTTRELL, AND R. K. BELEW, *Latent semantic indexing is an optimal special case of multidimensional scaling*, in Proc. Annual International ACM SIGIR Conf. Research and Development in Information Retrieval, ACM, 1992, pp. 161–167.
- [3] M. BELKIN AND P. NIYOGI, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation, 15 (2003), pp. 1373–1396.
- [4] M. BERRY AND M. BROWNE, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, SIAM, Philadelphia, 1999.
- [5] M. BERRY, Z. DRMAC, AND E. JESSUP, *Matrices, vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335–362.

- [6] M. BERRY, S. DUMAIS, AND G. O'BRIEN, *Using linear algebra for intelligent information retrieval*, SIAM Review, 37 (1995), pp. 573–595.
- [7] M. W. BERRY, *SVDPACK: A Fortran-77 Software Library for the Sparse Singular Value Decomposition*, Tech. Rep. CS-92-159, University of Tennessee, Knoxville, TN, June 1992.
- [8] M. W. BERRY, B. HENDRICKSON, AND P. RAGHAVAN, *Sparse matrix reordering schemes for browsing hypertext*, in Lectures in Applied Mathematics, vol. 32, AMS, 1996, pp. 99–123.
- [9] S. DEERWESTER, S. DUMAIS, G. FURNAS, T. LANDAUER, AND R. HARSHMAN, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41 (1990), pp. 391–407.
- [10] I. S. DHILLON, *Co-clustering documents and words using bipartite spectral graph partitioning*, in Proc. 7th Intl. Conf. Knowledge Discovery & Data Mining, ACM, 2001.
- [11] M. FIEDLER, *Algebraic connectivity of graphs*, Czech. Math. Journal, 23 (1973), pp. 298–305.
- [12] ———, *A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory*, Czech. Math. Journal, 25 (1975), pp. 619–633.
- [13] B. HENDRICKSON AND R. LELAND, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.
- [14] M. JUVAN AND B. MOHAR, *Optimal linear labelings and eigenvalues of graphs*, Discrete Appl Math., 36 (1992), pp. 153–168.
- [15] T. G. KOLDA, B. W. BADER, AND J. P. KENNY, *Higher-order web link analysis using multilinear algebra*, in Proc. IEEE Intl. Conf. Data Mining, November 2005.
- [16] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998.
- [17] M. C. NEWTON, O. SYKORA, AND I. VRTO, *Two new heuristics for the 2-sided bipartite crossing number*, in Proc. 10th Intl. Symp. Graph Drawing, Lecture Notes in Computer Science 2528, Springer, 2002, pp. 312–319.
- [18] G. SALTON AND M. MCGILL, *Introduction to Modern Information Retrieval*, McGraw Hill, New York, 1983.
- [19] D. A. SPIELMAN AND S.-H. TENG, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proc. ACM Symp. Theory of Computing (STOC), 2004, pp. 81–90.
- [20] K. STÜBEN, *A review of algebraic multigrid*, J. Computational & Applied Math., 128 (2001), pp. 281–309.
- [21] H. ZHA, X. HE, C. DING, M. GU, AND H. SIMON, *Bipartite graph partitioning and data clustering*, in Proc. 10th Intl. Conf. Info. & Knowledge Mgmt., ACM, 2001.