Taming Text with the SVD

Russ Albright, Ph.D.
SAS Institute Inc., Cary, NC
russell.albright@sas.com

January, 2004

Abstract

SAS Text Miner uses the vector space model for representing text. In this framework, distinct terms in the collection correspond to variables and documents represent observations. For most collections, the number of variables needed to represent each document is well above what can easily be modeled. As a result, dimension reduction becomes a crucial aspect of text mining solutions. In this paper, we explore the purpose and role of the Singular Value Decomposition (SVD) as a dimension reduction tool for text mining. We explain the mathematical foundation from which it is derived, provide an intuitive explanation of how it works, and provide guidence for using it in text mining applications. For those familiar with Principal Components Analysis (PCA), we include a discussion of the relationship of PCA to the SVD.

1 Introduction

Text Mining is an inherently difficult task. For the most part, programmatically processing text at a semantic level, by attempting to understand what the text is saying, has not been a realistic goal. Consider, for a moment, what your mind is doing in order to comprehend this document. What is communicated here is not entirely contained in the words on this page. Your knowledge and experience help to bring meaning to the text. For instance, when we spoke of "dimension reduction" in the abstract of this paper, you already had (we hope, anyway) a notion that we were talking about reducing the number of variables in a data set and not about decreasing your waist size. Since you are probably an analyst or statistician, or at least have a technical background, you bring the appropriate meaning to that phrase based on your experience.

Because of the complexity involved in understanding documents, only very specific domain applications can possibly tackle text at the semantic level. Instead, most text mining applications focus at the level of relationships between words in documents. With this approach, we try to learn what we can from the collection by examining the patterns of co-occurrence that exist across the collection.

In this paper we refer to collections, documents, and terms. *Collections* consist of any set of n documents. We use the words "documents" and "terms" loosely. Depending on the situation, a *document* may simply be a title, phrase, sentence, paragraph, or query, as well as what one typically thinks of as being a document. Depending on the level of tokenization, a *term* may have its typical connotation of simply being a "word" but may also refer to such items as punctuation, phrases or other multiword items.

Some of the examples in this document refer to a data set of technical support notes relating to various SAS products. These documents contain problems and suggested solutions for a variety of technical support issues. There were $10,440$ technical support notes contained in the collection. Each note is paragraph-sized containing between ten and several hundred terms. The following note is representative of the type of information found in the SAS Notes data set:

The following errors may be received when attempting to export graphics, such as when generating a GIF file: ERROR: Graphics device is offline. ERROR: Unable to open graphics device I/O. ERROR: Unable to initialize graphics device. ERROR: Driver SASGDGIF will not load. These errors can occur when the PROFILE and PROFILE2 catalogs have become corrupted. To correct the problem, exit SAS and delete or rename the PROFILE and PROFILE2 catalogs. The catalogs will be rebuilt when SAS is restarted, and the file should then export successfully.

	d1	**d2**	**d3**
1. error	1	1	1
2. invalid	1	0	0
3. message	1	2	0
4. file	1	1	0
5. format	1	0	1
6. unable	0	1	1
7. to	0	1	1
8. open	0	1	0
9. using	0	1	0
10. path	0	1	0
11. variable	0	0	1

Table 1: A Term-Document Frequency Matrix

1.1 Vector Space Models for Text

SAS Text Miner (TM) uses the vector space textual model introduced by Salton [7] for representing documents. Although the vector space model ignores the context of each word in a document (commonly referred to as the "bag of words" approach), it is useful because it provides an efficient, quantitative representation of each document. In this approach, documents are represented as vectors of length m, where m is the number of unique terms that are indexed in the collection. For any given document, the i^{th} entry of its vector representation is typically a function of the frequency of term i in that document multiplied by a weighting for the term. The weights can be generated with any of an assortment of functions that take into account the frequency of the term throughout the collection (see Yang and Pederson [9]). Although the weights can play a key role in the analysis, for this paper we will disregard them and restrict our discussion to where the i^{th} entry is simply the frequency of Term i in the document. The vector for each document is generally very sparse (i.e., it contains a high proportion of zeroes) because few of the terms in the collection as a whole are contained in any one given document.

If m is the number of distinct terms in a collection of n documents, then let A be the $m \times n$ matrix that represents this collection. This matrix, where terms are rows and documents are columns, is known as the term-document frequency matrix. As an example, consider this collection of three error messages where each message is considered to be a document:

1. Error: invalid message file format,
2. Error: Unable to open message file using message path,
3. Error: Unable to format variable.

Then the corresponding term-document frequency matrix is displayed in Table 1.

For SAS Text Miner, it is more natural to think about documents being rows (observations) and terms being columns (variables). As a result, for some of our discussion we will refer to the transpose of the term-document frequency matrix.

Another storage method for a term-document frequency matrix is displayed in Table 2. This is a compressed representation and is the actual representation used by the TM node. The magnitude of the compression is not evident in this example because the matrix has so few documents, but in general the compression allows for a significant savings in memory usage. With this approach only nonzero entries in the original matrix need to be stored. Typically this amounts to less than 5% of the entries in the uncompressed term-document frequency matrix. For communicating the ideas in this paper, however, we will refer to the uncompressed representation found in Table 1 (or its transpose) because the compressed

TERMNUM	_DOCUMENT_	_COUNT_
1	1	1
1	2	1
1	3	1
2	1	1
3	1	1
3	2	2
4	1	1
4	2	1
5	1	1
5	3	1
6	2	1
6	3	1
7	2	1
7	3	1
8	2	1
9	2	1
10	2	1
11	3	1

Table 2: Compressed Form of the Term-Document Frequency Matrix

matrix only adds a layer of confusion to the discussion.

In the matrix found in Table 1, there exists one variable for every term in the collection. Collections of even a few thousand paragraph-sized documents can easily contain tens of thousands of terms. This quickly presents a problem when building predictive models due to the "curse of dimensionality". As the collection size increases, the vector representation of a document becomes very sparse because very few of the distinct terms in the collection are actually contained in any single document.

Finally, the vector representation of documents allows for a simple measure of similarity. The dot product of any two document vectors indicates the strength of the similarity between the two documents and the angle between the two vectors can be used as a distance between the two documents. For example, the dot product of Document **d1** with Document **d2** gives

$$\begin{aligned} \mathbf{d1} \cdot \mathbf{d2} &= (1 \cdot 1) + (1 \cdot 2) + (1 \cdot 1) \\ &= 4 \end{aligned}$$

and the dot product of Document **d1** with Document **d3** is

$$\begin{aligned} \mathbf{d1} \cdot \mathbf{d3} &= (1 \cdot 1) + (1 \cdot 1) \\ &= 2 \end{aligned}$$

(for brievety, only nonzero addends are listed in the formulation of the dot product above). These results suggest, by pure term co-occurrence, that Document **d1** is more similar to Document **d2** than it is to Document **d3**.

1.2 Dimension Reduction of Documents

Suppose we had a collection of 20 documents, where each document could contain at most two words, *Word_A* and *Word_B* (A rather unusual collection!). Each document can be plotted based on the frequency of occurrence of each of the two words in the document. This is shown in Figure 1. Now suppose you were told that you could not use two dimensions to represent the documents. They must be represented in only one dimension so that every document must lie on a line. How would you do this?

Initially, you might decide to simply ignore one of the terms, say Term_B, and represent each document with only the frequency of Term_A. Graphically, this is depicted in Figure 2. The solid dots representing the documents have been projected onto the the Term-A-axis as circles. This approach, eliminating terms, can be accomplished in several ways in Text Miner and is discussed in Section 2.

On second thought, you may consider choosing a different line than one of the axes.[1] For instance, you could first draw a line through the points in such a way that the sum of the distances from each point

[1]Probably a better way to think of this is that instead of choosing a new line, you are going to choose a new axis for your coordinate system. Mathematically this can be done whenever you find a basis for the subspace.

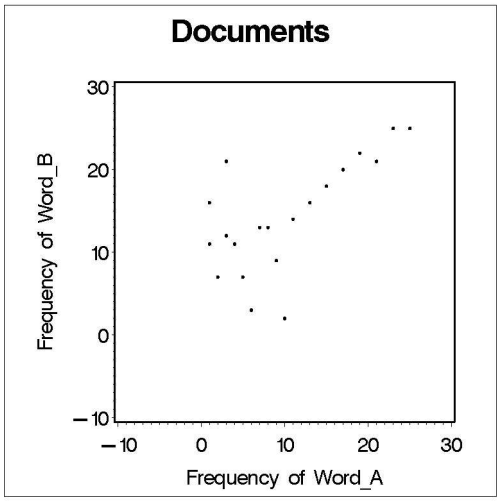


Figure 1: Scatter Plot of Documents

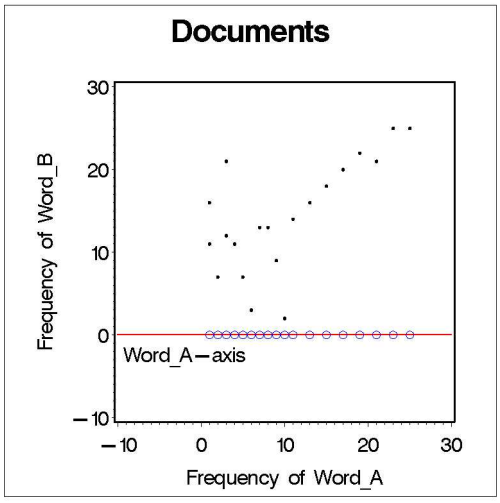


Figure 2: Eliminating Term_B

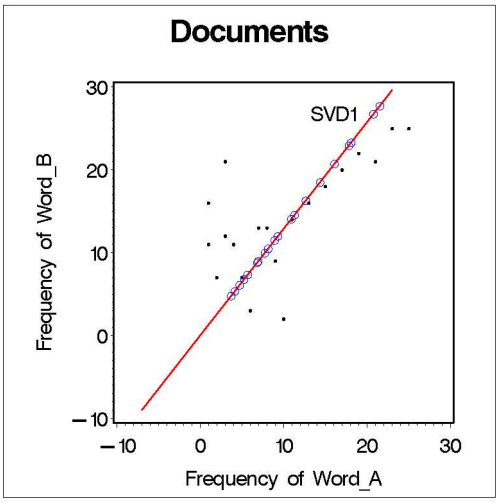


Figure 3: Projections onto the First Singular Vector.

to the line is minimized. This line is shown in Figure 3. The documents can then be perpendicularly projected onto this new line. The circles indicate the locations of the projected documents. This line and the new locations for the documents, can be obtained by using the SVD. The technique is used by default in Text Miner and it is discussed in Section 3.

Finally, you may think that the SVD line is not the best line to use after all. Instead, you may consider that having the points spread out as much as possible on this new line is what is best. This line is drawn in Figure 4. In this case you are maximizing the variance of the points that are projected onto the line. This line is formed by performing Principal Component Analysis as discussed in Section 4. The technique is not directly available in Text Miner but we include a discussion here because many readers are already familiar with it.

Which line is the best one to use? Well, that depends on the nature of your text, on the goals of your text mining analysis and on the computational resources you have available to you. In Section 5, we discuss these aspects, especially in relation to the SVD.

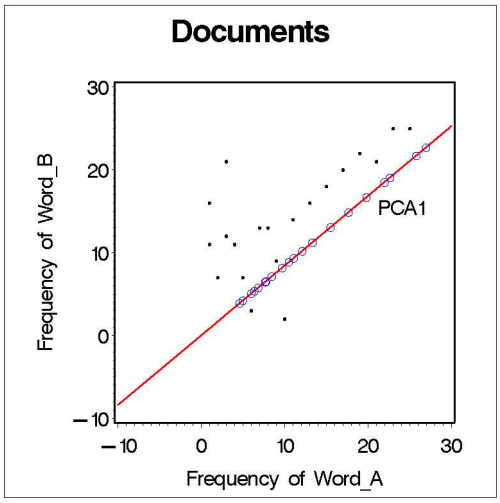


Figure 4: Projections onto the First Principal Component

2 Term Elimination as Dimension Reduction

SAS Text Miner provides many ways in which terms can be eliminated thereby reducing the dimension of a text mining problem. Since initially terms are variables in the data set, the most direct way to reduce the size of the document representation is to remove terms that are non-informative. Depending on the domain from which the text is derived and the goal of the text mining application, many terms can be omitted while enhancing the model. For most term elimination techniques, however, dimension reduction is only a secondary benefit. The primary effect is to reduce the noise in the data. The two exceptions to this are the *start list* and *roll-up terms* techniques. These provide significant dimension reduction. If the user does not apply a start list or does not use the "roll-up terms" technique then the SVD will almost certainly have to be applied in order to reduce the dimensionality before mining the text.

2.1 "Roll-Up Terms" and Start Lists

Text Miner offers two techniques that allow one to significantly reduce the number of terms kept:

1. Start Lists - keep only a fixed, predetermined set of terms for the analysis.

2. "Roll-Up Terms" - restrict the number of variables to the specified number of highest weighted terms.

Start lists are typically either formed by a user with domain knowledge about the problem or are discovered from a previous analysis of similar textual data. On the other hand, "roll-up terms" is a dynamic approach to reducing the dimensionality based on the weight assigned to each term in the *current* analysis.[2] Terms with weights that are below a given threshold will be removed from the analysis.

Both of the above techniques allow the user to eliminate most of the terms in the collection. As a result these techniques can reduce the dimensionality enough to be effective in predictive modeling.

2.2 Parsing Options Effect on Dimension Size

SAS Text Miner includes other techniques that have a side effect of reducing the dimension of a document vector:

1. Stop Lists - eliminate a fixed, predetermined set of terms from the analysis.

2. Synonym Lists - map pre-specified terms to a representative term.

3. Stemming - map multiple terms to a fixed term based on the root form.

[2] In TM "roll-up terms" actually serves two purposes, selecting the terms to use based on their weights and transforming the compressed representation to an uncompressed representation. In TM, no matter what term elimination techniques you apply, if you do not use the SVD, you must use "roll-up terms" to transform the compressed representation properly. Generally, before "roll-up terms" or the SVD is applied, any number of various other term elimination techniques would have been applied.

4. Remove Singly Occurring - drop terms that occur in no more than 1 document.

5. Remove Numeric Terms - drop terms that contain digits.

Yet other options in SAS Text Miner have the opposite effect, increasing the number of variables representing a document. These approaches involve including the following types of terms:

1. Punctuation

2. Entities

3. Noun phrases[3]

4. Part-of-speech tags.

Figure 5 conveys the amount of reduction that can be gained by choosing different Text Miner settings. The heights of the bars give a relative indication of the number of terms used in representing the collection of 10,440 paragraph-sized SAS technical support documents. The first bar in each pair indicates the number of distinct terms in the collection, regardless of the number of documents that the term occurred in. The second bar indicates the number of distinct terms that occurred when those that occurred in only one document are removed. The stop list used is the default stop list provided for English in TM containing 330 terms.

2.3 Problems with Text and Term Co-occurrence

Obviously, even the most aggressive settings still leave over ten thousand variables in the data set. This is far too many to analyze for a typical text mining solution. In addition, simply eliminating uninformative terms from the analysis does not address the issues of synonymy, polysemy, and term correlation that are identified by Deerwester *et al.* [2].

[3]TM includes the individual words of a noun phrase when the noun phrase option is selected. However, even without including those subterms, using noun phrases typically increases the number of distinct terms in the collection.

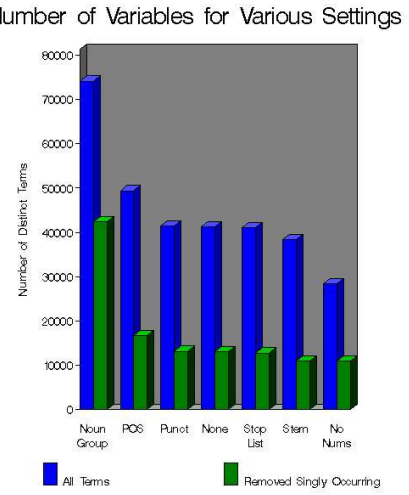


Figure 5: Term Counts by Settings

Synonymy refers to the characteristic of language to have several terms that mean essentially the same thing. In the SAS technical support data set the terms "frozen" and "hangs" often refer to the same situation where the program has reached a point where nothing is happening and yet the user cannot continue working. Using co-occurrence, the two terms represent separate variables and hence there is no indication based solely on either of those terms that the two documents may in fact be related.

On the opposite end of the spectrum from synonymy is polysemy. *Polysemy* is the tendency for the same term to mean different things in different contexts. The term "monitor" in technical support data is a good example of this. At times it refers to the computer screen, sometimes it refers to a piece of software that displays a graphical result and still other times it refers to the user "watching" or "observing" an event. In this case, many support notes discussing different issues will appear to be similar because only the term itself, not the term's meaning, is used to characterize the variable.[4]

[4]One approach for dealing with this problem is to "tag" the term with a part of speech that is based on its usage. Sometimes identical terms can then be distinguished by considering the tag. If the various meanings for a given term corresponds to a different part of speech (and your part-of-speech tagger

Term dependence refers to the tendency for certain terms to be highly correlated with one another. This problem is not unique to text but also occurs with most other sets of data as well. The terms "error" and "message" are strongly correlated in the technical support collection. When one occurs, the other also tends to occur. A pair of documents, each containing these two terms, may have their similarity overrated in this case.

The above problems are simply not addressed when the user uses only terms as variables. Text Miner provides the SVD as an alternative because it helps to resolve some of these problems caused by using terms as variables and at the same time, it accomplishes the obvious need to reduce the number of variables that are used to represent the document.

3 Dimension Reduction with the SVD

The singular value decomposition is an important matrix factorization technique that plays a primary part in many linear algebra algorithms and has a significant role in conceptually understanding many concepts in linear algebra (see Trefehen and Bau [8]). In order to effectively use the SVD, it is helpful to understand some key concepts discussed in this section.

3.1 The SVD Applied to the Term-Document Frequency Matrix

The SVD is a matrix factorization method for both real and complex rectangular matrices. The details of the SVD can be found in many matrix analysis texts. Meyer [5] includes a clear introduction with an application to term-document frequency matrices. Deerwester *et al.* [2] were the first to apply the SVD to a term-document frequency matrix. This was done in the context of information retrieval, not text mining, and in that context is referred to as Latent Semantic Indexing (LSI).[5]

The SVD comes in two forms, a full SVD, and a reduced SVD. In this paper, we only discuss the reduced representation of the decomposition as conveyed by Trefethen and Bau [8].[6] For our purposes, we will restrict our discussion to the term-document frequency matrix. This matrix has real nonnegative entries and is sparse. However, while we will focus on this particular matrix, much of what is stated here generalizes to matrices that do not necessarily have these particular properties.

3.1.1 Definition

Let A be an $m \times n$ term-document frequency matrix with rank r, $r \leq n$. Without loss of generality let $m \geq n$ hold so that there are more terms than documents. The singular value decomposition of A can be stated succinctly as

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

where U is an $m \times r$ orthogonal[7] matrix whose columns make up the *left singular vectors*, Σ is an $r \times r$ dimensional diagonal matrix whose diagonal elements are termed *singular values*, and V is an $r \times n$ orthogonal matrix whose columns form the *right singular vectors* of A. The diagram in Figure 6 gives a schematic representation.

The singular values are all greater than or equal to zero and, by convention, are ordered from largest to smallest, with the largest one placed in the upper left corner of the matrix Σ. In the reduced representation, all of the singular values in Σ will be greater than zero. The singular values themselves help to characterize some of the properties of A and this is discussed further in Subsection 5.2.

is accurate on your particular text) then this approach can be effective.

[5] TM uses the svd technique in order to discover patterns across the collection of documents. LSI, on the other hand, is designed to return a subset of documents that are related to a specific query.

[6] The reduced and full factorization produce equivalent results. In practice, only the reduced representation is used because it eliminates the need for storing some unnecessary rows and columns of matrices.

[7] A real matrix M with orthonormal columns is called an *orthogonal matrix* and has the unusual property $M^{-1} = M^T$. *Orthonormal* vectors have unit length and are mutually orthogonal to one another.

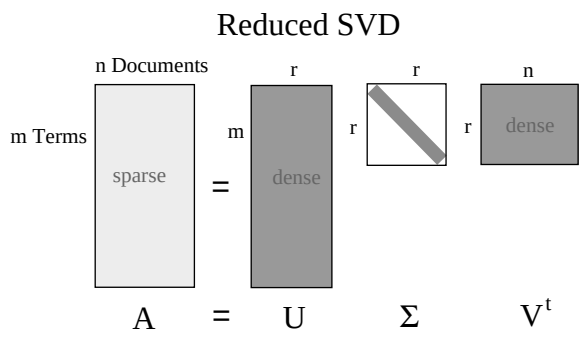


Figure 6: The SVD for a Term-Document Frequency Matrix

3.1.2 Approximations with the SVD

The power behind Text Miner's use of the SVD comes from an important property of the SVD regarding approximating matrices. The property is best described by noting that the SVD can be viewed as a sum of rank one matrices:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T.$$

The matrix A can then be approximated by choosing any $k \leq r$. This generates a rank k matrix, A_k, that is the best rank-k approximation to A in terms of least-squares best fit:

$$A \approx A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^t = U_k \Sigma_k V_k^t \qquad (1)$$

The diagram in Figure 7 demonstrates the approximation. Note that the k in Figure 6 is significantly smaller than the r in Figure 7.

Equation 1 implies that one can get a rough approximation to A by taking the product of the first singular value with the matrix formed from the outer product of the first column of U with the first column of V. The matrix formed, A_1, will be $m \times n$ but will be of only rank one. Of all possible matrices, B, of rank one, $||A - B||_2$ will be smallest when $B = A_1$ holds. One can improve the approximation by forming the product of the second singular value with the outer product of the second columns of U and V, and then adding this result to A_1. The resultant matrix, A_2, will be the the best rank-two approximation to A. The approximations can be successively improved by repeating the process until $k = r$ holds and the original matrix is produced.

Geometrically, the effect of this approximation was shown in Figure 3. In this example, the underlying term document frequency matrix is a 2×20, rank-two matrix. Since it already has only rank of two, the only approximation is for $k = 1$. The subspace is merely a line, and the documents are projected onto this line.

As a result, the value of k controls how good an approximation one has achieved. If $k = r$ holds, then A equals A_k is satisfied. Again, since the desire is to reduce the dimensions significantly, typically k will be chosen to be much smaller than r. We discuss this further in Section 5.2.

The approximation from Equation 1 can be visually demonstrated by using gray-scale to convey the magnitude of the entries of A as we do in Figure 8. The plot represents the term-document frequency matrix of a artificial collection of 100 documents and 500 terms. The gray-scale in the plot conveys the frequency of the i^{th} term occurring in the j^{th} document. The matrix was constructed so that term i could occur as few as 0 times in Document j (the ij^{th} region is completely white) or as many as 4 times (the ij^{th} region is black).[8]

The terms and documents have been purposely arranged in the matrix so that there is a relatively clear grouping between many of the documents. Note that in practice, a typical collection could not be arranged with such a clarity between clusters of documents and terms. In fact, if one could arrange the rows and columns in such a way, there would be little need to actually run a clustering algorithm; the clusters are evident from the structure of the matrix.

In Figure 9 we display several approximations to the original term-document frequency matrix by using Equation 1 and varying values of k. The first four matrix plots of Figure 9 are rank-one through

[8]The proc used to generate this plot, PROC GCONTOUR, actually averages the shade of gray for each region based on the shades of gray of those areas surrounding it.

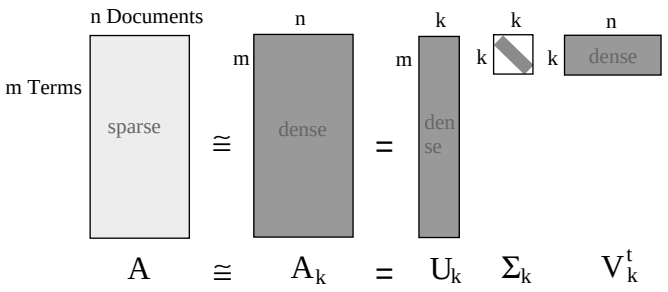


Figure 7: Approximating the Term-Document Frequency Matrix

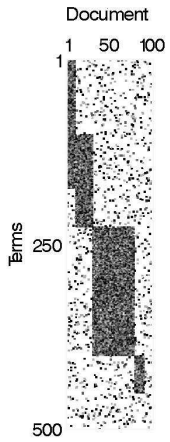


Figure 8: Matrix Plot

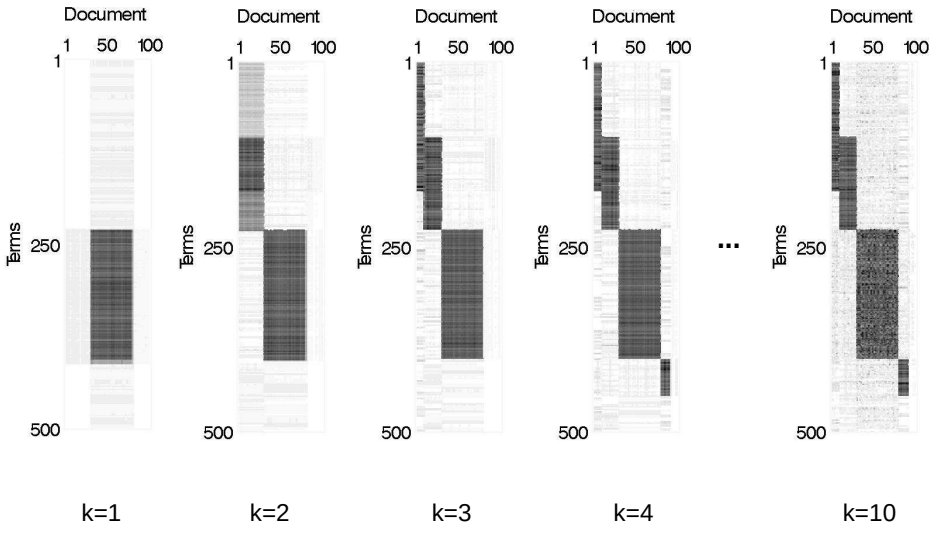


Figure 9: A_k for Varying Values of k

rank-four approximations. The last plot is a rank-ten approximation. One can see that with only a rank-one approximation the largest "cluster" in the original matrix is evident. Rank-two and rank-three approximations successively reveal additional structure. The rank-four approximation reveals all of the main structure and, finally, a rank-ten plot shows an even finer level of granularity. In this example, only the four most significant singular triples are needed to capture the essence of the matrix due to its block structure. For real, non-contrived examples, things are not as nice and clear cut.

3.1.3 The SVD and the Eigenvalue Decomposition

There is a notable connection between the singular value decomposition of A and an eigenvalue decomposition of either A^TA or AA^T. This connection is of interest because it relates Text Miner's use of the SVD to Principal Component Analysis and is discussed further in Section 4. For now we will merely state the mathematical relationship between the two decompositions.

Let B be an $n \times n$ matrix of real values. The eigenvalue decomposition of B is a factorization of the form

$$B = X\Lambda X^{-1}, \tag{2}$$

where X is an $n \times n$ matrix whose columns are known as eigenvectors and where Λ is an $n \times n$ diagonal matrix of eigenvalues. Note that this factorization only holds when the eigenvectors are linearly independent and hence, X^{-1} exists.

If the matrix $B = A^TA$ (or $B = AA^T$) is formed from a rectangular matrix A, then the nonzero singular values of A are the square root of the nonzero eigenvalues of B. Further, the eigenvectors of A^TA are the right-hand singular vectors for A and the eigenvectors of AA^T are the left-hand singular vectors of A. This result is straightforward to show:

$$\begin{aligned} AA^T &= \left(U\Sigma V^T\right)\left(V\Sigma U^T\right) \\ &= U\Sigma\left(V^TV\right)\Sigma U^T \\ &= U\Sigma I\Sigma U^T \end{aligned}$$

$$= U\Sigma^2 U^T.$$

As a result, one way to calculate the SVD of a matrix is to first form AA^T and then calculate the eigenvalues and eigenvectors for this product.[9] The square roots of the eigenvalues of A^TA are the singular values of A and the eigenvectors of A^TA are the left singular vectors of A. Forming the product the other way, with A^TA, will yield the right singular vectors of A.

3.2 Documents in k-Dimensional Space

Of course, the ultimate goal of applying the SVD is to place documents (or terms) in a much lower dimensional space than that defined by the term space (or document space). Given the factorization in Equation 1, a k-dimensional space is available. The two matrices, U_k and V_k, form an orthonormal basis for the k-dimensional document and term spaces respectively. That is, their columns can represent a new set of axes for a k-dimensional space.

An m-dimensional document $\mathbf{d}$ that is represented as a vector of terms, can be projected on to this lower dimensional subspace with

$$\widehat{\mathbf{d}} = U_k^T \mathbf{d}. \tag{3}$$

So $\widehat{\mathbf{d}}$ is a k-dimensional vector whose i^{th} entry is formed by taking a linear combination of the original term frequencies with the ith singular vector (column) of U_k. The entries in the columns of U_k can be viewed as weights to be applied to the individual terms in the collection. The projection given in Equation 3 is remarkably simple because the columns of U_k already form an orthonormal basis of the k-dimensional space.

In Figure 3 we have already shown the geometry of the SVD projections. As a computational example, however, consider the term-document frequency matrix from Table 1. Let $k = 2$ hold. Then the truncated SVD decomposition of A is given by

$$A \approx U_2 \Sigma_2 V_2^T$$

and the values for U_2, Σ_2, and V_2 (rounded to the nearest hundredth) are:

$$U_2 = \begin{bmatrix} .43 & .30 \\ .11 & .13 \\ .55 & -.37 \\ .33 & -.12 \\ .21 & .55 \\ .31 & .18 \\ .31 & .18 \\ .22 & -.25 \\ .22 & -.25 \\ .22 & -.25 \\ .09 & .42 \end{bmatrix},$$

$$\Sigma_2 = \begin{bmatrix} 3.79 & 0 \\ 0 & 1.96 \end{bmatrix},$$

and

$$V_2 = \begin{bmatrix} .43 & .25 \\ .82 & -.49 \\ .36 & .83 \end{bmatrix}.$$

The columns of U_2 are the singular vectors. These vectors are orthogonal to one another and hence form a basis for the reduced space.

Now we project the first document consisting of the terms

Error: invalid message file format

into 2-dimensional space. This document is represented as

$$\mathbf{d} = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]^T$$

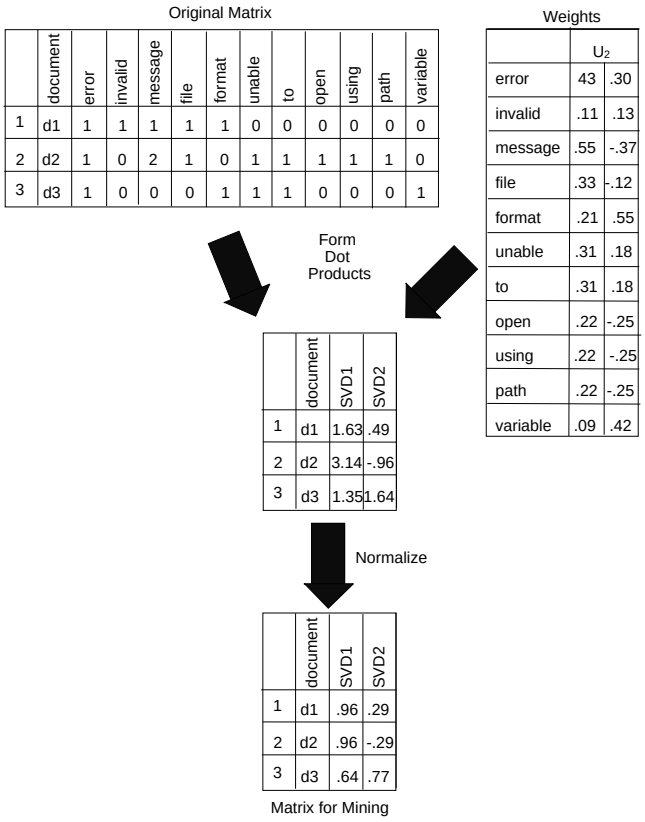
in our vector space model and becomes

$$\widehat{\mathbf{d}} = U_2^T \mathbf{d} = [1.63, 49]^T$$

after applying the SVD projection. Text miner actually normalizes the above result (placing the documents on the unit hypersphere) to get

$$\widehat{\mathbf{d}}_{normalized} = [.96, .29]^T$$

[9] The algorithm TM uses is actually based on this approach to finding the components of the SVD.



	document	error	invalid	message	file	format	unable	to	open	using	path	variable
1	d1	1	1	1	1	1	0	0	0	0	0	0
2	d2	1	0	2	1	0	1	1	1	1	1	0
3	d3	1	0	0	0	1	1	1	0	0	0	1

	U_2	
error	43	.30
invalid	.11	.13
message	.55	-.37
file	.33	-.12
format	.21	.55
unable	.31	.18
to	.31	.18
open	.22	-.25
using	.22	-.25
path	.22	-.25
variable	.09	.42

	document	SVD1	SVD2
1	d1	1.63	.49
2	d2	3.14	-.96
3	d3	1.35	1.64

	document	SVD1	SVD2
1	d1	.96	.29
2	d2	.96	-.29
3	d3	.64	.77

Figure 10: Transformation of the Data Set.

so that euclidean distance can be used to compare documents rather than the angle between the documents.

Similarly the other two documents can also be projected into this subspace. The end result, after taking the transpose, is in the bottom table contained in Figure 10. The figure also communicates the significant reduction that has been accomplished in representing the documents. The 11 original variables have been replaced with 2 variables by way of the SVD. The two variables are each linear combinations (prior to normalization) of the original 11 with weights applied from the U matrix of the SVD.

If you think of terms as being observations and documents as being variables (similar to what is depicted in Table 1) then terms can also be projected into a reduced space. Although not demonstrated here, the matrix V_2 , for instance, allows one to represent terms in only two dimensions in a way that corresponds to Equations 3.

4 PCA and the SVD

Principal Components Analysis (PCA) is a well-known multivariate statistical technique for transforming a number of correlated variables into a few uncorrelated variables. The technique was invented by Pearson [6] and expanded on by Hotelling [3]. If you are already familiar with the technique, the discussion in the previous section of the various aspects of the SVD may have reminded you of PCA.

4.1 Description of PCA

PCA is useful for arbitrary rectangular matrices just as the SVD is, but in keeping with the tone of the paper, we will again restrict our discussion to the $m \times n$ term-document frequency matrix A. From the rectangular matrix A, PCA specifies that the square covariance or correlation matrix[10] C is formed and then the eigenvalue decomposition of C is calculated:

$$C = X\Lambda X^{-1}.$$

As described in Subsection 3.1.3, the eigenvalue decomposition produces the eigenvalues Λ and eigenvectors X necessary for projecting the original document into the reduced space just as was done in Subsection 3.2.

Those that have developed and used PCA have established terminology for the various aspects of the technique. The eigenvectors that are used to form the coefficients for the equation of the line as shown in Figure 4 are known as *principal components.* The associated eigenvalues are called *principal values.* The i^{th} principal value is proportional to the additional variance described by adding the i^{th} principal component. As a result, the ratio

$$v_i = \frac{\lambda_i}{\sum_{j=1}^{r} \lambda_j} \tag{4}$$

indicates the variation captured by the i^{th} component. A plot of the principal values called a *scree plot* is often used to visualize the amount of the variance explained by using only k principal components. This is demonstrated in Section 5.2. Finally, the individual elements of the principal component vectors are frequently known as *loadings* and forming the multiplication as in Equation 3 is known as applying the loadings to form the *principal component scores.*

4.2 Comparing the Approaches

Although based on equivalent procedures, since PCA and TM's SVD approach operate on different data, they do not produce the same results. Depending on whether the raw data is used or the covariance matrix is used, different vectors will be found as basis vectors for the reduced space.[11]

In Figure 11, the first principal component from PCA is compared to that of the SVD approach.

[10] Recall that the covariance matrix of A can be formed by subtracting the mean of each column of A from each entry in that column to form B. Then the covariance matrix C is merely $1/\sqrt{m-1}B^T B$, where m is the number of rows in the matrix. The correlation matrix is similar except it standardizes the variance between each of the columns. The correlation matrix is typically used when the variables represent measurements based on differing scales. For the term-document frequency matrix, this is not the case.

[11] This was first made evident to the author from an unpublished document written by Oliver Dane in October 5, 1999.

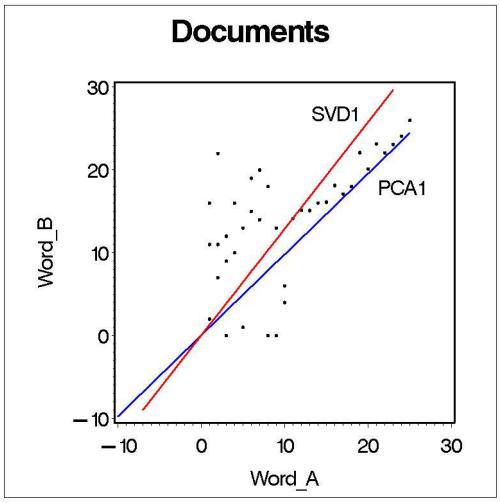


Figure 11: First Principal Component for the Two Approaches

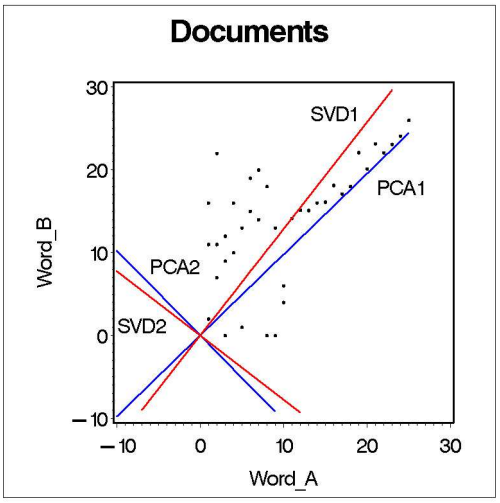


Figure 12: First and Second Principal Components for the Two Approaches

While the PCA component maximizes the variance, the SVD finds the best fitting line in the least-squares sense. The SVD line is affected by the dispersement of Word_B when Word_A occurs between 0 and 10 times while the PCA line is not influenced by this secondary variation. Depending on the nature of the data, these two lines may or may not be fairly close to one another. If Text Miner were to use the mean-adjusted term-document frequency data, rather than the raw data, the SVD approach and PCA, applied to the covariance matrix, would produce identical results.

In Figure 12 the second principal component and singular vector are drawn. In both cases, the second component is orthogonal to the first component. Note that, with the two components, the complete space is captured, and the "projection" of each document places it in the same location. The only difference is that a different coordinate system (either the two SVD or the two PCA lines serve as the new axes) is now used to represent its location.

Text Miner does not use the covariance or correlation matrix, or even form the mean-adjusted matrix for a very good reason: the resulting matrix would no longer be sparse. The dimensions of A are so large, that taking advantage of a compressed representation as given in Table 2 is a requirement. Forming the mean-adjusted matrix would not allow this.

The variance criterion of the principal values from Equation 4 also holds in relation to the SVD. That is, the square of the singular values quantifies the amount of additional variance explained by adding the i^{th} singular vector:

$$v_i = \frac{\sigma_i^2}{\sum_{j=1}^{r} \sigma_j^2}. \tag{5}$$

Text Miner provides a plot of these singular values from the interactive results. However, a decision to use k dimensions based on the proportion

$$p_k = \frac{\sum_{i=1}^{k} \sigma^2}{\sum_{j=1}^{r} \sigma_j^2} \tag{6}$$

is not as applicable for the SVD as it is for PCA because the criteria for selecting each additional component using the SVD approach is not based on maximizing the explained variance as it is for PCA.

5 Practical Considerations for the Use of the SVD in Text Mining

Several questions arise when using the SVD as a dimension reduction tool. In this section, we address three of these questions:

- When is the SVD technique appropriate?
- How can one effectively choose the number of SVD dimensions? and
- How can one interpret the new variables that are created by the SVD?

5.1 When is the SVD Technique Appropriate?

For most Text Mining problems, the SVD will be entirely appropriate to use. Without a data reduction technique, there will be more variables (terms) available than one can use in a data mining model. Some method must be applied to select an appropriate set from which a text mining solution can be built. Unlike term elimination, the SVD technique allows one to derive significantly fewer variables from the original variables.

There are some drawbacks to using the SVD, however. Computationally, the SVD is fairly resource intensive and requires a large amount of RAM. The user must have access to these resources in order for the decomposition to be obtained.

Since the SVD dimensions are linear combinations of the input terms, sometimes a blurring can occur between documents that are distinguishable when using the terms themselves as variables. For instance, using the SVD, Albright *et al.* [1] had difficulties distinguishing between documents about "corn" and "wheat" even though a simple term-based classifier works quite well in separating the two types of documents. The problem arises when the document collection (which in this case was based on a varied collection of financial-related documents) is diverse but the discrimination is desired at a fine level. Sometimes indexing only the "right" subset of terms, with a start list or "roll-up terms", can give very strong separation between two clusters/categories when the SVD cannot.

TM's SVD approach is not well-suited for textual data that does not have a rich interaction between terms and documents. For instance, analyzing documents that only consisted of author names with the SVD would not be reasonable. There is too little co-occurrence between terms (the individual first and last names) contained in the documents (first and last names taken together). The rank of such a term-document frequency matrix will be very small and if the SVD can even be computed on such a matrix, many documents will likely be placed at the origin of the reduced subspace, because they will contain no terms that have a nonzero weight.

5.2 How Many Dimensions Should be Used?

The choice for the number of dimensions k to use can be a crucial aspect of many text mining solutions. With too few dimensions, the model will fail to explain prominent relationships in the text. On the other hand, using too many dimensions will add unnecessary noise to the model and make training an effective model nearly impossible. In practice, there is an upper bound of at most a few hundred dimensions from which to build a model. So the user should not need to consider more than this.

Choosing the optimal number for any given text mining problem is an active area of research [1, 2] producing few clear guidelines. In our text mining experience, most cases perform best when using between 10 and 250 dimensions, However, the precise number to use needs to be learned through experimentation and varies with the text, the TM settings, and the goal of the text mining activity.

In some rare cases it may be possible to make the choice based on the plot of the singular values. With PCA, this is known as a scree plot. If the scree plot indicates that the rate of change of squared singular values is beginning to level out, this is an indication that enough singular values have been kept. For instance, Figure 13 shows that after $k = 4$, little additional explanation of variance is gained. This

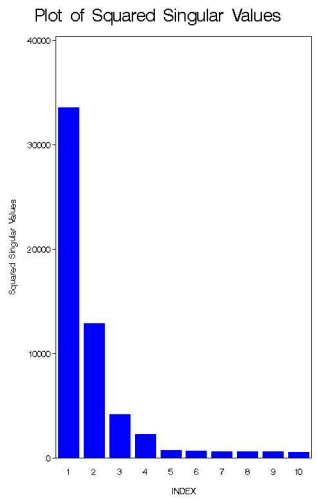


Figure 13: Plot of Squared Singular Values Corresponding to Figure 8

corresponds to the results shown in Figure 9 in which the plot corresponding to $k = 4$ is nearly as effective as the plot corresponding to $k = 10$.

Some have suggested calculating the total variance of all the data and then stopping when a certain percentage, S, of that variance has been reached by using the sum of the first k $v_i's$ given in equation 5. While this is possible, the number of dimensions needed to explain a pre-specified percentage of the variance can vary greatly based on the document collection, parsing settings, and desired percentage to be explained. At times, even though the first few singular values are much larger than later singular values, the cumulative effect of the later singular values can dwarf the contribution of the first couple of hundred singular values.

This is not always the case, however. In the collection of $10,440$ technical support documents containing over $10,000$ terms, 75% of the variance in the collection is accounted for with the first 73 dimensions. Accounting for 90% of the variance requires 467 dimensions. At times, a compromise must be obtained between the number of variables that are practical in a text mining model and the desired amount of variance to be explained. TM currently does not facilitate the use of the comparison with the total variance, but it is likely that TM will make use of these ideas in the future to help the user select the appropriate number of dimensions.

5.3 How can One Interpret the Dimensions?

Another common desire is to provide an interpretation of the new orthogonal singular vectors. Recall again from Figure 10 how these new dimensions are formed. The matrix, U_K consists of weights that are applied to the entries in the term-document frequency matrix. The individual weight suggests by its value how "important" the corresponding term is for the given SVD dimension. For instance, in Figure 10, since the weight of the term "message" is significantly higher in the first column of U_2 than the second column, it could be one of the descriptive terms for the first dimension. If there are several terms with a higher weight for a given dimension, and those terms correspond to a single concept, then the concept itself could be used as a label for the given dimension.

Many textbook descriptions of PCA provide an example in which these dimensions can be interpreted easily. In almost all of these cases, less than 20 dimensions are mapped to 2 or 3 dimensions. In this case, it is possible to make reasonable judgments about what those dimensions mean. For text mining, the problem of interpreting dimensions is magnified by the sheer number of input variables. Tens of thousands of variables are mapped to fewer than one hundred variables, which makes interpretation extremely difficult.

As a result, TM only uses the SVD for the dimension reduction aspect. Any interpretation is done at the end of the process by analyzing the frequencies of the terms that occur within a given cluster or category.

6 Conclusions

SAS Text Miner uses the vector space model for representing text. We have shown that this model produces tens of thousands of variables for even moder-

ately sized problems. Since each distinct term in the collection becomes a variable, there are simply too many for practical use. Users of Text Miner have a choice between reducing the number of variables by eliminating terms from the representation or replacing the term variables with new variables that are linear combinations of those original variables (possibly after eliminating some of them).

The latter technique is the SVD approach. We have discussed how the approach chooses a new set of dimensions and projects the document vectors down to this reduced space. These dimensions become the new variables in the SAS data set. The benefit to the user is that documents are now represented with an effective number of dimensions and that the new variables are less correlated. The disadvantage is that the dimensions are no longer easily interpreted and the precise number of dimensions to choose must be determined through experimentation. Determining the optimal number of dimensions is a topic of ongoing research.

Finally, we have shown that TM's SVD approach and PCA can be formalized as computationally identical techniques. However, since PCA operates on the mean adjusted matrix, it chooses orthogonal dimensions that optimally account for the variance in the data. The SVD approach, operating on the raw data, simply selects orthogonal dimensions that reduce the sum of the perpendicular distances from the original observations to the new axis.

Acknowledgements

The author would like to thank Ross Bettinger, James Cox, Bernd Drewes, Annette Sanders, Leslie Warren, and Terry Woodfield of SAS Institute for their review of drafts of this paper. The author would also like to express appreciation to Amy Langville and Carl Meyer of North Carolina State University for their constructive comments, corrections, and interaction regarding this article.

References

[1] R. Albright, J. A. Cox, K. Daly. Skinning the Cat: Comparing Alternative Text Mining Algorithms for Categorization, Proceedings of the 2nd Data Mining Conference of DiaMondSUG, Chicago, IL. DM Paper 113, 2001.

[2] 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, pp. 391-407, 1990.

[3] Hotelling, H. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24, 417-441, 1933

[4] I. T. Jolliffe, *Pincipal component analysis*. Springer, New York, 1986.

[5] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000.

[6] K. Pearson, On lines and planes of closest fit to systems of points in space. Philosophical Magazine, 2, 559-572, 1901.

[7] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

[8] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[9] Y. Yang and J. Pedersen, A comparative study on feature selection in text categorization. In Machine Learning: Proceedings of the Fourteenth International Conference (ICML97), pp. 412-420, 1997