

Using Vector Spaces for Information Retrieval

Greg Brown and Liya Zhu

December 14, 2001

frithsplot@yahoo.com

[Problems with Classic ...](#)

[Bridging the Gap](#)

[Geometry of the Vector ...](#)

[Comparison](#)

[Rank Reduction](#)

[QR Factorization](#)

[An Example of QR ...](#)

[Comparison using QR ...](#)

[More Geometry](#)

[The Low Rank ...](#)

[Singular Value ...](#)

[An Example](#)

[The Reduced Rank ...](#)

[Term-Term Comparison](#)

[Conclusion](#)

[Home Page](#)

[Title Page](#)

[◀](#) [▶](#)

[◀](#) [▶](#)

Page 1 of 25

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

Abstract

The digitization of data has provided new challenges in storing and indexing information. As the Internet grows, massive amounts of data need to be efficiently indexed. One method for accomplishing this uses the power of vector spaces. The purpose of this paper is to implement information retrieval with the vector space model. Then we will explore more advanced applications of linear algebra to information retrieval and finally follow with a practical example.

[Problems with Classic...](#)

[Bridging the Gap](#)

[Geometry of the Vector...](#)

[Comparison](#)

[Rank Reduction](#)

[QR Factorization](#)

[An Example of QR...](#)

[Comparison using QR...](#)

[More Geometry](#)

[The Low Rank...](#)

[Singular Value...](#)

[An Example](#)

[The Reduced Rank...](#)

[Term-Term Comparison](#)

[Conclusion](#)

[Home Page](#)

[Title Page](#)

◀

▶

◀

▶

Page 2 of 25

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

- [Problems with Classic . . .](#)
- [Bridging the Gap](#)
- [Geometry of the Vector . . .](#)
- [Comparison](#)
- [Rank Reduction](#)
- [QR Factorization](#)
- [An Example of QR . . .](#)
- [Comparison using QR . . .](#)
- [More Geometry](#)
- [The Low Rank . . .](#)
- [Singular Value . . .](#)
- [An Example](#)
- [The Reduced Rank . . .](#)
- [Term-Term Comparison](#)
- [Conclusion](#)

1. Problems with Classic Indexing Systems

Most early twentieth century libraries used the card catalog as a standard for indexing information. This is effective for a library with, say, 10,000 documents. However, even the most intrepid librarian would run into trouble attempting to index the roughly 2 billion web pages that currently make up the Internet. For this we need a new method of indexing. This new method has become the science of information retrieval. Through the use of computers and a bit of applied mathematics, we will show how a large database of information can be precisely and efficiently indexed.

2. The Vector Space Model

The first problem presented by attempting to bridge the gap between text and mathematics is how to represent a document with numbers. There are many methods for doing this. In this paper we will represent each document as a vector. The elements of each vector represent the frequency of a certain term in that document. Consider the following example.

Let us use the title “The Art of Computer Programming” as a document. The terms “art,” “computer,” and “mathematics” will be used to index the document. Each element will represent the frequency with which each of these terms appear in the document. The term-document vector becomes

$$v = (1 \quad 1 \quad 0)^T$$

This representation has some obvious flaws. “art,” in the common sense of the word, has very little relevance to computer programming. However, it receives the same numerical importance as “computer,” a term that is more relevant to this document. This fundamental problem with our model will be dealt with as we explore more advanced methods of information retrieval.

[Home Page](#)

[Title Page](#)

◀◀ ▶▶

◀ ▶

Page 3 of 25

[Go Back](#)

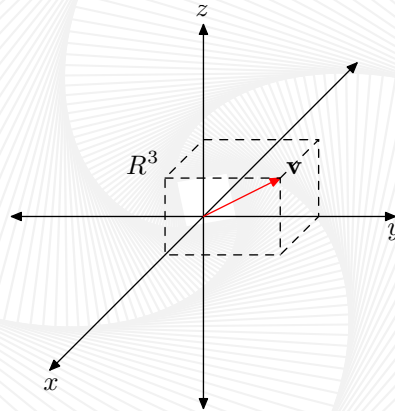
[Full Screen](#)

[Close](#)

[Quit](#)

3. Geometry of the Vector Space

We can now think of a document as a vector. The next step is to create a geometrical analogue. A n dimensional vector will be represented in space. Here is our example vector, $v = (1 \ 1 \ 0)^T$, graphed in 3 dimensional space:



Now we need a way to compare two vectors. For example, if a user queries for the term “computers,” there must be a way to compare the relevance of a document to this query. The query will be represented in n dimensional space as a vector. For our example the vector is

$$q = (0 \ 1 \ 0)^T.$$

4. Comparison

We now need a method of comparison between a query and a document vector. A common method of comparison is to find the cosine between the vectors:

$$\cos \theta = \frac{a_j^T q}{\|a_j\| \|q\|}.$$

Notice that scaling a_j or q will not change the value of the cosine. Now we can expand this vector interpretation to use many documents. A database with d documents and t terms will become a $t \times d$ document matrix. Then each element in a column is the relevancy of a predefined term to the document. Thus the element A_{ij} is a numerical representation of the relevancy term i has to document j . This method of comparison is geometrical. We do not compare the vectors literally, but through geometry.

There are many options for representing the relevancy of a term to a document. A common method is to use the frequency of the term to the document. We will apply this method to the following example.

The terms of the matrix:

T_1 : compu(ters, ational,ting,ter)

T_2 : programming

T_3 : mathematics

T_4 : algebra(ic)

T_5 : algorithms

T_6 : cryptography

The documents we are describing:

D_1 : The Art of Computer Programming

D_2 : Computer-Aided Mathematics Programming with Matlab

D_3 : Computational Mathematics

D_4 : Algebraic Computations with Computers

D_5 : Computer Algorithms for Cryptography

Now create a 6×5 term-by-document matrix. The columns will represent each document.

The elements of the matrix represent the frequency that each term occurs in the document.

$$A^* = \begin{pmatrix} T_1D_1 & T_1D_2 & T_1D_3 & T_1D_4 & T_1D_5 \\ T_2D_1 & T_2D_2 & T_2D_3 & T_2D_4 & T_2D_5 \\ T_3D_1 & T_3D_2 & T_3D_3 & T_3D_4 & T_3D_5 \\ T_4D_1 & T_4D_2 & T_4D_3 & T_4D_4 & T_4D_5 \\ T_5D_1 & T_5D_2 & T_5D_3 & T_5D_4 & T_5D_5 \\ T_6D_1 & T_6D_2 & T_6D_3 & T_6D_4 & T_6D_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Since the content of a document is determined by the relative frequencies of the terms and not

by total number of times they appear, we now normalize the matrix:

$$A = \begin{pmatrix} .7071 & .5774 & .7071 & .7071 & .5774 \\ .7071 & .5774 & 0 & 0 & 0 \\ 0 & .5774 & .7071 & 0 & 0 \\ 0 & 0 & 0 & .7071 & 0 \\ 0 & 0 & 0 & 0 & .5774 \\ 0 & 0 & 0 & 0 & .5774 \end{pmatrix}$$

Now we have a matrix with each column representing a specific document. To compare the query vector with each document, find the cosine between each column in the matrix and the query vector. As an example say a user queries for “programming.” This gives the query vector:

$$q^{(1)} = [0 \ 1 \ 0 \ 0 \ 0 \ 1]^T.$$

The cosines between each document vector and the query vector give respective values of .5000, .4082, 0, 0, .4082. If we use a cutoff of .5, the query will return the first document in the set. Unfortunately the second document is an excellent reference as well, yet it is not returned from this query. This is a second fundamental problem with information retrieval. We will deal with this later, using more advanced mathematical techniques.

5. Rank Reduction

The problem with information retrieval in the real world occurs when matrices become large. A real world example might have millions of entries. In order to make searching through these matrices practical, we must find a way to reduce the number of entries. This is the true power of information retrieval. The remainder of this paper will be concerned with reducing the rank of a matrix.

In an extremely large matrix many dependencies can be expected. When this matrix is reduced, we can expect a great deal of zero entries. The lower the rank of the matrix, the more zero entries that can be expected. These entries can be ignored in future computations. Therefore, it is advantageous to reduce the rank of the matrix as much as possible, while retaining its integrity.

6. QR Factorization

To begin understanding the concept of rank reduction, one must understand how a dependence between the columns of A results in reducing the rank of A . A simple way to find the basis for the column space of a matrix is to compute the QR factorization:

$$A = QR.$$

This will create a matrix Q with all orthonormal columns. The matrix R will be upper triangular. We must show that A is a subset of the column space of Q . This will allow us to see dependence among the columns of A by examining the matrix Q .

$$[a_1 \ a_2 \ \dots \ a_n] = [q_1 \ q_2 \ \dots \ q_n] \begin{bmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_n^T \end{bmatrix}$$

$$[a_1 \ a_2 \ \dots \ a_n] = [q_1 r_1^T \ q_2 r_2^T \ \dots \ q_n r_n^T]$$

Then A is a linear combination of the columns of Q . Then, any column in A can be written as a linear combination of the columns of Q . If we can show that Q has linear dependence, it follows that A must have the same linear dependence.

7. An Example of QR Factorization

Now for an illustration of QR factorization. We will use the example matrix from before.

$$\left(\begin{array}{cccc|cccc} -0.71 & 0 & 0.71 & 0 & 0 & 0 & 0 & 0 \\ -0.71 & -0 & -0.71 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.71 & -0.71 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.71 & 0.71 & 0 & 0 \end{array} \right) \left(\begin{array}{ccccc} -1 & -0.82 & -0.50 & -0.50 & -0.41 \\ 0 & -0.58 & -0.71 & 0 & 0 \\ 0 & 0 & 0.50 & 0.50 & 0.41 \\ 0 & 0 & 0 & 0.71 & 0 \\ 0 & 0 & 0 & 0 & -0.82 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right) \quad (1)$$

Many ideas can be gathered from this numerical factorization. In order to show that the basis of A is contained in the independent columns of Q , we use the following block multiplication:

$$\begin{aligned} A &= (Q_A \quad Q_A^\perp) \begin{pmatrix} R_A \\ 0 \end{pmatrix} \\ &= Q_A R_A + Q_A^\perp * 0 = Q_A R_A \end{aligned} \quad (2)$$

This block multiplication makes it quite clear that the columns in Q_A^\perp have no effect on the column space of A . Then the columns in Q_A make up a basis for the column space of A . The alert reader will note the partitioning above required the use of a permutation matrix:

$$PA = QR.$$

However, this only serves to reorder the document vectors. It has no actual effect on the data in the database. Henceforth we will refer to the factorization $PA = QR$ as $A = QR$.

8. Comparison using QR Factorization

We have now shown that the matrix Q_A provides a basis for the column space of A . Let us substitute the factored form of A into the earlier comparison equation:

$$\begin{aligned}\cos \theta_j &= \frac{a_j^T q}{\|a_j\| \|q\|} \\ &= \frac{(Q_A r_j)^T q}{\|Q_A r_j\| \|q\|}\end{aligned}\tag{3}$$

Next we use the remarkable properties of an orthogonal matrix to simplify this equation.

$$\|Q_A r_j\| = \sqrt{(Q_A r_j)^T Q_A r_j} = \sqrt{r_j^T Q_A^T Q_A r_j} = \sqrt{r_j^T I r_j} = \sqrt{r_j^T r_j} = \|r_j\|$$

Thus the cosine formula reduces to:

$$\cos \theta_j = \frac{r_j^T (Q_A^T q)}{\|r_j\| \|q\|}$$

To show this produces the same result, we query against the same vector as earlier. This makes cosines of .5000, .4082, 0, 0, .4082. This is identical to our previous example.

9. More Geometry

The QR factorization allows us a new geometrical interpretation of the vector space. We now need to think in terms of projections. First remember

$$I = Q Q^T = (Q_A \quad Q_A^\perp) (Q_A \quad Q_A^\perp)^T = Q_A Q_A^T + Q_A^\perp (Q_A^\perp)^T.$$

[Home Page](#)
[Title Page](#)


Page 10 of 25

[Go Back](#)
[Full Screen](#)
[Close](#)
[Quit](#)

Problems with Classic...
Bridging the Gap
Geometry of the Vector...
Comparison
Rank Reduction
QR Factorization
An Example of QR...
Comparison using QR...
More Geometry
The Low Rank...
Singular Value...
An Example
The Reduced Rank...
Term-Term Comparison
Conclusion

Now use the properties of orthogonal matrices to craft the following:

$$\begin{aligned}
 q &= Iq \\
 &= QQ^T q \\
 &= [Q_A Q_A^T + Q_A^\perp (Q_A^\perp)^T] q \\
 &= Q_A Q_A^T q + Q_A^\perp (Q_A^\perp)^T q.
 \end{aligned} \tag{4}$$

The important part occurs now. Recall that Q_A is a basis for the column space of A and QQ^T is the projection matrix onto the column space of Q . Then

$$Q_A Q_A^T q + Q_A^\perp (Q_A^\perp)^T q = q_A + q_A^\perp \tag{5}$$

where q_A and q_A^\perp are the projections of q onto the spaces of Q_A and Q_A^\perp respectively. Therefore, the properties of the projection allow us to say that q_A is the best approximation of the query vector in the column space of A .

Substitute the result from [Equation 5](#) into [Equation 3](#). This produces

$$\begin{aligned}
 \cos \theta_j &= \frac{a_j^T (q_A + q_A^\perp)}{\|a_j\| \|q\|} \\
 &= \frac{a_j^T q_A + a_j^T q_A^\perp}{\|a_j\| \|q\|}
 \end{aligned} \tag{6}$$

Recall from [Equation 5](#)

$$q_A^\perp = Q_A^\perp (Q_A^\perp)^T q.$$

Substituting this into [Equation 6](#) creates

$$\cos \theta_j = \frac{a_j^T q_A + a_j^T Q_A^\perp (Q_A^\perp)^T q}{\|a_j\| \|q\|}$$

Note that a_j is in the column space of A , which is an orthogonal complement to the column space of Q_A^\perp . Then $a_j^T Q_A^\perp = 0$. Then the formula becomes

$$\begin{aligned}\cos \theta_j &= \frac{a_j^T q_A + 0 * (Q_A^\perp)^T q}{\|a_j\| \|q\|} \\ &= \frac{a_j^T q_A}{\|a_j\| \|q\|}\end{aligned}\quad (7)$$

Essentially the query from the user has been replaced by its most relevant counterpart from the column space of A . This is q_A . Then taking the above equation a little farther

$$\cos \theta'_j = \frac{a_j^T q_A}{\|a_j\| \|q_A\|}.$$

Now we are using the projection of the query to compare with each document. Examining a comparison between these two different methods of searching leads to

$$\cos \theta_j = \cos \theta'_j \frac{\|q_A\|}{\|q\|}$$

Recall from earlier that q_A is the projection of q onto the column space of A . From [Equation 4](#) and [Equation 5](#) we have $q_A^\perp = q - q_A$. Then from the geometry of the projection we can use the Pythagorean Theorem to say

$$\cos \theta'_j = \frac{\|q_A\|}{\sqrt{\|q_A\|^2 + \|q_A^\perp\|^2}}.$$

The value $\sqrt{\|q_A\|^2 + \|q_A^\perp\|^2}$ is obviously larger than $\|q_A\|$. Then the cosines computed using q_A will be greater than those using q . Since these cosines will be greater in magnitude, they are

more likely to be returned as relevant. This new formula will return more documents. These documents might be relevant, which is good, or they may be irrelevant, which is bad. Either way this method has the potential to better the search.

10. The Low Rank Approximation

The geometrical insight provided by the QR factorization is quite enlightening. However, it has other applications. As said at the beginning of this paper, a primary trouble with database indexing lies in its inherent uncertainty. One possible solution to this is to create a matrix of uncertainty, say matrix E . Therefore the matrix $A + E$ will represent a new matrix \hat{A} . If used properly, this will create a more exact approximation of what is actually represented in the database than our original matrix. To begin this analysis we need an important definition:

$$\|X\|_F = \sqrt{\sum_{i=1}^t \sum_{j=1}^d x_{ij}^2}.$$

This is called the Frobenius matrix norm. Think of it as an analogue to the Euclidean norm for vectors, a way in which to measure the magnitude of a matrix. It can also be shown that

$$\|X\|_F = \sqrt{\text{Trace}(X^T X)} = \sqrt{\text{Trace}(X X^T)}$$

which leads to multiplying X by a $n \times n$ orthogonal matrix O

$$\|OX\|_F = \sqrt{\text{Trace}((OX)^T(OX))} = \sqrt{\text{Trace}(X^T O^T O X)} = \sqrt{\text{Trace}(X^T I X)} = \|X\|_F.$$

Similarly for an $m \times m$ matrix V

$$\begin{aligned}
 \|XV\|_F &= \sqrt{\text{Trace}((XV)^T(XV))} \\
 &= \sqrt{\text{Trace}((XV)(XV)^T)} \\
 &= \sqrt{\text{Trace}(XVV^T X^T)} \\
 &= \sqrt{\text{Trace}(XIX^T)} \\
 &= \|X\|_F.
 \end{aligned} \tag{8}$$

Using these new tools, we will now try to find the new matrix \hat{A} . From the QR factorization we have the matrix R , which also has the rank of A . Looking closer we can create a new matrix \hat{R} by changing the last pivot row to all zeros. This creates a low rank approximation to R .

$$R = \left(\begin{array}{ccc|cc}
 -1 & -0.82 & -0.50 & -0.50 & -0.41 \\
 0 & -0.58 & -0.71 & 0 & 0 \\
 0 & 0 & 0.50 & 0.50 & 0.41 \\
 0 & 0 & 0 & 0.71 & 0 \\
 \hline
 0 & 0 & 0 & 0 & -0.82 \\
 0 & 0 & 0 & 0 & 0
 \end{array} \right) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \tag{9}$$

we can see what happens when the matrix is approximated by removing the first nonzero row from the bottom. Note that this only changes the size of the matrix by:

$$\|R_{22}\|_F / \|R\|_F = .3658. \tag{10}$$

Then we can create the new matrix \hat{R} by setting R_{22} to the zero matrix. This new matrix will have a rank of 4. Then the new matrix $A + E$ is different from the original matrix A by E .

Then

$$\begin{aligned}
E &= (A + E) - A \\
&= Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} - Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \\
&= Q \begin{pmatrix} 0 & 0 \\ 0 & -R_{22} \end{pmatrix}
\end{aligned} \tag{11}$$

From [Equation 8](#)

$$\|E\|_F = \left\| \begin{pmatrix} 0 & 0 \\ 0 & -R_{22} \end{pmatrix} \right\|_F = \|R_{22}\|_F$$

Now we need a way to compare $\|A\|_F$ to $\|\hat{A}\|_F$ to observe the difference created by lowering the rank.

$$\|A\|_F = \|QR\|_F = \|R\|_F \tag{12}$$

Since $E = R_{22}$

$$\frac{\|E\|_F}{\|A\|_F} = \frac{\|R_{22}\|_F}{\|R\|_F} = .3658 \tag{13}$$

It follows that there will be approximately a 36% change in the size of R when we reduce the rank by 1. Since $\|A\|_F = \|R\|_F$, a 36% change in R corresponds to the same change in A. This may appear to be a large change, but running the query against this new reduced vector returns cosines of .5000, .4082, 0, 0, 0, 0. These are extremely close to the values returned from the original matrix.

11. Singular Value Decomposition

The QR factorization can be helpful in increasing the precision and efficiency of information retrieval techniques. From that factorization we can create a reduced rank basis for the column

space of A . Now we can refine our technique to return even more relevant results. As mentioned earlier a fundamental problem with search engines occurs with what is known as polysemy. For example, think of the word run. Run could be contained in the document “running monopolistic corporations” or “running from hungry grizzly bears.” These documents contain the same term but are not related in any obvious manner. Therefore we need a way with which to compare terms to terms, thusly refining our result to more relevant documents. In order to compare terms to terms we need information about the row space of A . Even better would be a reduced rank approximation. This is accomplished through the use of the Singular Value Decomposition or the SVD:

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

U is a $t \times t$ orthogonal matrix containing what are called the left singular values of A as its columns. V is the $d \times d$ orthogonal matrix that contains the right singular values of A as its columns. Σ is a $m \times n$ diagonal matrix that contains the singular values of A along its diagonal

$$\begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 & 0 \\ & & \ddots & & & \\ 0 & 0 & 0 & \sigma_r & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (14)$$

where $\sigma_{i-1} > \sigma_i$. We can use this factorization for any matrix A . There is important information in this factorization. First

$$\text{rank}(A) = \text{rank}(U\Sigma V^T).$$

Since V^T is invertible

$$\text{rank}(A) = \text{rank}(U\Sigma)$$

and since U is invertible as well

$$\text{rank}(A) = \text{rank}(\Sigma).$$

Next recall [Equation 8](#) and

$$\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma V^T\|_F = \|\Sigma\|_F.$$

Since Σ is a diagonal matrix

$$\|\Sigma\|_F = \sqrt{\sum_{j=1}^{\text{rank}(A)} \sigma_j^2}$$

The QR factorization and the SVD have many similarities. The rank of A , or r_A , is the same as the rank of Σ just as it is the same as the rank of R . The first r_A vectors in Q are a basis for the column space of A just as the first r_A vectors in U . In addition, the first r_A rows in V^T are a basis for the row space of A . Similar to the reduction method we used for QR factorization we can create a r_k approximation to the A where $r_k \leq r_A$. In order to do so, we set all but the k largest singular values of A to zero. A primary difference between the QR factorization and the SVD is the accuracy with which they reduce the rank of the matrix A . For the SVD this accuracy is measured by the difference between the matrix A and the r_k approximation A_k :

$$\|A - A_k\|_F = \|A - X\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_{\text{rank}(A)}^2} \quad (15)$$

Here $A_k = U_k \Sigma_k V_k^T$. U_k is the $t \times k$ matrix whose columns are made up of the first k columns of U , V_k is the $d \times k$ matrix whose columns are made up of the first k columns of V , and Σ_k is the $k \times k$ diagonal matrix whose diagonal elements are the k largest singular values of A .

12. An Example

Now we can try this new method of factorization on our first example. Recall

$$A = \begin{pmatrix} .7071 & .5774 & .7071 & .7071 & .5774 \\ .7071 & .5774 & 0 & 0 & 0 \\ 0 & .5774 & .7071 & 0 & 0 \\ 0 & 0 & 0 & .7071 & 0 \\ 0 & 0 & 0 & 0 & .5774 \\ 0 & 0 & 0 & 0 & .5774 \end{pmatrix}$$

[Problems with Classic ...](#)

[Bridging the Gap](#)

[Geometry of the Vector ...](#)

[Comparison](#)

[Rank Reduction](#)

[QR Factorization](#)

[An Example of QR ...](#)

[Comparison using QR ...](#)

[More Geometry](#)

[The Low Rank ...](#)

[Singular Value ...](#)

[An Example](#)

[The Reduced Rank ...](#)

[Term-Term Comparison](#)

[Conclusion](#)

[Home Page](#)

[Title Page](#)



Page 18 of 25

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

Then using the SVD

$$\begin{aligned}
 U &= \begin{pmatrix} -0.83 & -0.20 & 0.12 & -0 & 0.51 & 0 \\ -0.36 & 0.45 & -0.16 & -0.71 & -0.38 & -0 \\ -0.36 & 0.45 & -0.16 & 0.71 & -0.38 & -0 \\ -0.16 & -0.32 & 0.75 & 0 & -0.55 & 0 \\ -0.11 & -0.48 & -0.43 & -0 & -0.27 & -0.71 \\ -0.11 & -0.48 & -0.43 & -0 & -0.27 & 0.71 \end{pmatrix} \\
 \Sigma &= \begin{pmatrix} 1.76 & 0 & 0 & 0 & 0 \\ 0 & 0.90 & 0 & 0 & 0 \\ 0 & 0 & 0.76 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0 \\ 0 & 0 & 0 & 0 & 0.20 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
 V &= \begin{pmatrix} -0.48 & 0.20 & -0.04 & -0.71 & 0.48 \\ -0.51 & 0.45 & -0.16 & 0 & -0.71 \\ -0.48 & 0.20 & -0.04 & 0.71 & 0.48 \\ -0.40 & -0.41 & 0.81 & 0 & -0.15 \\ -0.35 & -0.74 & -0.57 & -0 & -0.09 \end{pmatrix}
 \end{aligned} \tag{16}$$

r_A is equal to the rank of Σ . Since the matrix Σ has two zero rows, it clearly follows that A is not full rank. In order to create a r_k approximation to A , where $r_k \leq r_A$, we keep k pivot rows in Σ and change the remaining rows to zero. For example, if we want to reduce A to rank 4, we can say that

$$\|A - A_4\| = \sqrt{\sigma_5^2} = \sigma_5 = .2 \tag{17}$$

Then the change from matrix A to the new matrix A_4

$$\frac{\|A - A_4\|_F}{\|A\|_F} = .1124 \quad (18)$$

According to the above result, only a 11% relative change is created when A is changed from a rank 5 matrix to rank 4. Compare this to the 36% change from the QR factorization. The SVD certainly has its advantages! We can further reduce the rank of matrix A to 3 by setting the fourth row of Σ to zeros. Similarly,

$$\|A - A_3\|_F = \sqrt{\sigma_4^2} = \sigma_4 = .71 \quad (19)$$

Then the relative change when changing A from rank 5 to rank 3 is

$$\frac{\|A - A_3\|_F}{\|A\|_F} = .4045 \quad (20)$$

Thus changing the rank of the matrix to 3 results in a 40% change to the matrix. This is a fairly significant change, so it appears that reducing the matrix to rank 4 is the best idea. The rank 4 approximation arrived at above is a 4 dimensional subspace of the original 5 dimensional matrix A . We then represent our database with its rank 4 approximation. One obvious problem is how to tell when we have gone too far in rank reduction. Unfortunately, there is no general rule here, where to stop is usually found by empirical experimentation. Before examining more properties of the SVD it is enlightening to look at the rank reduced approximations computed by the different factorizations.

The original matrix:

$$A = \begin{pmatrix} .7071 & .5774 & .7071 & .7071 & .5774 \\ .7071 & .5774 & 0 & 0 & 0 \\ 0 & .5774 & .7071 & 0 & 0 \\ 0 & 0 & 0 & .7071 & 0 \\ 0 & 0 & 0 & 0 & .5774 \\ 0 & 0 & 0 & 0 & .5774 \end{pmatrix}$$

The rank 4 approximation from the QR factorization:

$$\hat{A} = \begin{pmatrix} 0.7071 & 0.5774 & 0.7071 & 0.7071 & 0.5774 \\ 0.7071 & 0.5774 & -0.0000 & 0 & 0 \\ 0 & 0.5774 & 0.7071 & 0.0000 & 0.0000 \\ 0 & 0 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The rank 4 approximation from the SVD:

$$A_4 = \begin{pmatrix} 0.6589 & 0.6494 & 0.6589 & 0.7227 & 0.5867 \\ 0.7427 & 0.5242 & 0.0356 & -0.0115 & -0.0069 \\ 0.0356 & 0.5242 & 0.7427 & -0.0115 & -0.0069 \\ 0.0522 & -0.0781 & 0.0522 & 0.6902 & -0.0101 \\ 0.0256 & -0.0382 & 0.0256 & -0.0083 & 0.5724 \\ 0.0256 & -0.0382 & 0.0256 & -0.0083 & 0.5724 \end{pmatrix}$$

From these figures it certainly looks like the QR factorization is a far superior approximation. However, we have just shown that the SVD is about 25% more precise. It is natural to ask how

this can possibly occur. At this point one must remember we are looking at the *geometrical* interpretation of these vectors. Recall that we use the cosine of the angle between vectors for comparison. This is not the same as the numerical representation of the vector.

13. The Reduced Rank Vector Space Model

Now that we can rank reduce using the SVD we need a method of query matching. We now interpret the vector space model in terms of the SVD. This is accomplished by comparing the query vector q to the approximation A_k of A . Now define e_j to be the j th vector of the $d \times d$ identity matrix. Then $A_k e_j$ will extract the j th column vector from A_k . Then we can compare these vectors to the original query vector.

$$\cos \theta_j = \frac{(A_k e_j)^T q}{\|A_k e_j\| \|q\|} = \frac{(U_k \Sigma_k V_k^T e_j)^T q}{\|U_k \Sigma_k V_k^T e_j\| \|q\|} = \frac{e_j^T V_k \Sigma_k (U_k^T q)}{\|\Sigma_k V_k^T e_j\| \|q\|} \tag{21}$$

This equation can be simplified by making the substitution $s_j = \Sigma_k V_k^T e_j$ and the formula reduces to

$$\cos \theta_j = \frac{s_j^T (U_k^T q)}{\|s_j\| \|q\|}.$$

The cosines can be computed without explicitly forming the $t \times d$ matrix A_k . This saves a great deal in computation time.

14. Term-Term Comparison

So far, we've applied the vector space model for comparing queries with documents. With minor variation, the model can also be used to do term-term comparison, which helps to refine

the results of a search automatically. Specifically, this will allow us to combat the problem of polysemy. For example, let us say a user queries a large document database for the term **interest**. This polysemous query will likely return documents that are in no way related to what the user desires. Take the following example:

The terms:

- T1: interest(ing)
- T2: money
- T3: hobbies
- T4: dividend
- T5: investment
- T6: curiosity
- T7: concern

The documents:

- D1: Rational Self Interest in American Society
- D2: Using Bank Interest on Investments to Create Dividends
- D3: Hobbies, Interests, and Curiosities for the Modern Man
- D4: Making Money in Interesting Ways
- D5: Concern for the Interests, Hobbies and Investments of America's Youth

[Problems with Classic ...](#)

[Bridging the Gap](#)

[Geometry of the Vector ...](#)

[Comparison](#)

[Rank Reduction](#)

[QR Factorization](#)

[An Example of QR ...](#)

[Comparison using QR ...](#)

[More Geometry](#)

[The Low Rank ...](#)

[Singular Value ...](#)

[An Example](#)

[The Reduced Rank ...](#)

[Term-Term Comparison](#)

[Conclusion](#)

[Home Page](#)

[Title Page](#)



Page 23 of 25

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

Then our 7×5 term-by-document matrix with unit columns becomes

$$G = \begin{pmatrix} 1 & 0.58 & 0.50 & 0.71 & 0.50 \\ 0 & 0 & 0 & 0.71 & 0 \\ 0 & 0 & 0.50 & 0 & 0.50 \\ 0 & 0.58 & 0 & 0 & 0 \\ 0 & 0.58 & 0 & 0 & 0.50 \\ 0 & 0 & 0.50 & 0 & 0 \\ 0 & 0 & 0.50 & 0 & 0.50 \end{pmatrix} \quad (22)$$

Now to compare each term to each other term, we use the cosine between each of the term vectors:

$$\cos \omega_{ij} = \frac{(e_i^T G)(G^T e_j)}{\|G^T e_i\| \|G^T e_j\|}, i, j = 1, \dots, 7 \quad (23)$$

e_i and e_j represent the i th column of the $t \times t$ identity matrix. We then put the cosines in the matrix C .

$$C = \begin{pmatrix} 1 & 0.46 & 0.46 & 0.38 & 0.50 & 0.33 & 0.46 \\ 0.46 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.46 & 0 & 1 & 0 & 0.46 & 0.71 & 1 \\ 0.38 & 0 & 0 & 1 & 0.76 & 0 & 0 \\ 0.50 & 0 & 0.46 & 0.76 & 1 & 0 & 0.46 \\ 0.33 & 0 & 0.71 & 0 & 0 & 1 & 0.71 \\ 0.46 & 0 & 1 & 0 & 0.46 & 0.71 & 1 \end{pmatrix} \quad (24)$$

$C_{ij} = \cos \omega_{ij}$. The greater the magnitude of the cosine, the better a match between the rows i and j . Once again we must think geometrically. If the magnitude of the cosine is large, the vectors are located near one another in the row space of A (the term-by-document matrix). This is where the SVD becomes extremely helpful.

Looking at the above matrix we can make partitions based on the relative geometry of each vector. It appears that columns 2, 4, and 5 are related. Column 3, 6, and 7 look related as well. Column 1 is just the term “interest” which is our polysemous term to begin with and is contained in every document. This analysis corresponds with the relation of the original document vectors to one another. The terms that are related to one another in a linguistic manner are now also related numerically. This method is known as clustering. More often it is implemented automatically to avoid human error.

Now the use of the SVD becomes obvious. We can create a reduced rank approximation to the column space using the matrix U . We can create a reduced rank approximation to the row space using V^T . Now we substitute the approximation into [Equation 23](#) to craft:

$$\cos \omega_{ij} = \frac{(e_i^T U_k \Sigma_k V^T)(V_k \Sigma_k U_k^T e_j)}{\|V_k \Sigma_k U_k^T e_i\| \|V_k \Sigma_k U_k^T e_j\|} = \frac{(e_i^T U_k \Sigma_k)(\Sigma_k U_k^T e_j)}{\|\Sigma_k U_k^T e_i\| \|\Sigma_k U_k^T e_j\|} \quad (25)$$

The true power of the SVD shines. We now have ways to compare terms to terms and terms to documents, all while increasing the relevancy of the search results.

15. Conclusion

We have truly come far in our understanding of search engines. With the vector space model a graphical representation of documents was created. This allowed us to apply mathematical methods for a more efficient and precise method of information retrieval. However, this does not begin to scratch the surface of the power of mathematics in relation to linear algebra. Hundreds of applications have turned this into a fascinating a prosperous industry.

References

- [1] Arnold, Dave *Mathematics Expertise*
- [2] Berry, Michael W. *Matrices, Vector Spaces, and Information Retrieval*
- [3] Berry, Michael W. *Understanding Search Engines*
- [4] Saucedo, Douglas *L^AT_EX Expertise*
- [5] Strang, Gilbert *Introduction to Linear Algebra*
- [6] Stewart, G.W. *Computer Science and Applied Mathematics*

[Problems with Classic...](#)

[Bridging the Gap](#)

[Geometry of the Vector...](#)

[Comparison](#)

[Rank Reduction](#)

[QR Factorization](#)

[An Example of QR...](#)

[Comparison using QR...](#)

[More Geometry](#)

[The Low Rank...](#)

[Singular Value...](#)

[An Example](#)

[The Reduced Rank...](#)

[Term-Term Comparison](#)

[Conclusion](#)

[Home Page](#)

[Title Page](#)



Page 26 of 25

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)