POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Matrices, Vector-Spaces and Information Retrieval

K. Ming Leung

Abstract: Recently developed information retrieval technologies are based on the concept of a vector space. Techniques from linear algebra can be used to manage and index large text collections.

Directory

- **Table of Contents**
- **Begin Article**

Last Revision Date: November 27, 2004

Table of Contents

1. Introduction

Two recommended sources of background material on IR systems are the textbooks by Frakes and Baeza-Yates[1] and Kowalski[2]. Both of these books are used in undergraduate and graduate courses in IR, and both provide good references on the design and performance of IR systems.

Recently developed information retrieval (IR)[3] technologies are based on the concept of a vector space. Data are modeled as a matrix, and a user's query of the database is represented as a vector. Relevant documents in the database are then identified via simple vector operations. Orthogonal factorizations of the matrix provide mechanisms for handling uncertainty in the database itself. Mathematical techniques from linear algebra can be used to manage and index large text collections.

2. Challenges in Indexing Large Collection of Information

A. Capacity or scale

1. Periodicals worldwide
   - i. 160,000 already in print
   - ii. 12,000 being added per year
2. Books in the U.S. alone:
   - i. 1.4 million already in print
   - ii. 60,000 being added per year

B. Indexing Inconsistencies

1 Indexing starts by selecting a list of ”terms” to be used to classify documents for a given database.

2 Extraction of concepts and keywords from documents for indexing is intrinsically a fragile process.

3 An average of 20% disparity in the terms chosen as appropriate to describe a given document by any 2 different professional indexers.

4 Selecting such a list by indexer depends on the experience and opinions of the indexer such as:

   - i. age
   - ii. cultural background
   - iii. education

iv. political orientation

v. language - languages have quite a bit of ambiguities due to polysemy (words having multiple meanings) and synonymy (multiple words having the same meaning).

For example in the English language, there are many words that have multiple meanings (polysemy). Like the word "bank" can be used to refer to the place where one goes to deposit or withdraw money. It is also used in phrases such as bank-shoot in billiard, banking of a road, or a bank of computer memory. In addition, multiple words or phrases can have pretty much the same meaning (synonymy). For example, go to the toilet, use the can, take a leak, take a pee, take a piss, take a whiz, make water, pass water, spend a penny, micturate, piddle, puddle, wee-wee, and relieve oneself all mean pretty much the same thing, *i.e.* to urinate.

Indexing performances are measured by

recall the ratio of the number of relevant documents retrieved to the total number of documents in the collection.

precision the ratio of the number of relevant documents retrieved to the total number of documents retrieved.

Standardized evaluation of IR began in 1992.

3. Vector Space Model

3.1. A Vector Space Formulation of Information

Suppose we are interested in a collection of documents and have a list of keywords or terms that we want to use to index or categorize each of these documents.

In the vector space model, associated with each document is a vector, called the document vector. Value of each element reflects the importance of a term in representing the semantics of the document. Typically the value is a function of the frequency with which the term appears in the document.

A database containing d documents described by t terms is represented as $t \times d$ term-by-document matrix, $\mathbf{A}$. The d vectors representing the d document form the columns of $\mathbf{A}$. The rows are the term vectors. Matrix element a_{ij} is the weighted frequency at which term i appears in document j.

The semantic content of the database is entirely contained in the column space of $\mathbf{A}$. Geometric relationship between document vectors is used to model similarities and differences in content. Term vectors are compared geometrically to identity similarities and differences in term usage.

Since a document generally uses a small subset of the entire dictionary of terms generated for the db, each document vector has many zeroes (sparse).

A user queries the db to find relevant documents. The query is represented by a $t \times 1$ vector, $\mathbf{q}$, just like each document vectors, with entries weighted according to the importance of the corresponding term in the query.

Query matching is finding document most similar (closest in some geometric sense) in use and weighting of terms.

A common measure of closeness is obtained from the cosine of the angle between the query and the document vectors. If matrix $\mathbf{A}$ has columns, $\mathbf{a}_j, j = 1, 2, \cdots, d$,

$$\cos\theta_j = \frac{\mathbf{a}_j^T\mathbf{q}}{\|\mathbf{a}_j\|_2\|\mathbf{q}\|_2}.$$

Note that $\|\mathbf{a}_j\|_2$ need to be computed only once for any given db. Also multiplying $\mathbf{q}$ or $\mathbf{a}_j$ but a constant does not change the value of the value of $\cos\theta_j$.

The larger $\cos\theta_j$ is the more similar are the query and the jth document vector.

3.2. A Simple Example

Let us consider a very simple example to illustrate the basic ideas. Suppose we are interested in books on the recipes for baking bread and pastries. We assume that we have a total of 6 terms to be used to classify these books. The terms are:

t	Terms
1	bak(e,ing)
2	recipe(s)
3	bread(s)
4	cake(s)
5	pastr(y,ies)
5	pie(s)

We are only interested in the root of each of these terms, so that for example the words bake and baking have the same root. Ways to strip away unnecessary endings of a given word to extract out the root are available and we assume that they have been used here.

Our collection of books has a total of 5 books. For the sake of simplicity (so that we do not need to show the entire text of each book), we assume that the semantic content of these books is captured by their respective titles. Their titles are:

d	Documents (Titles)
1	How to Bake Bread without Recipes
2	The Classic Art of Viennese Pastry
3	Numerical Recipes: The Art of Scientific Computing
4	Breads, Pastries, Pies and Cakes: Quality Baking Recipes
5	Pastry: A Book of Best French Recipes

Therefore we have $d = 5$ and $t = 6$. Normally d is much much larger than t. The $t \times d$ term-by-document matrix before normalization has element $\hat{a}_{ij}$ is the number of times term i appears in the document-title j. We find that

$$\hat{\mathbf{A}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

After normalizing each columns using the Euclidean norm, the term-by-document matrix is given to 4 significant figures by

$$\mathbf{A} = \begin{bmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{bmatrix}.$$

Now for example a user initiates a search for books about baking bread. The query vector is then given by

$$\mathbf{q}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We compute $\cos\theta_j$ between $\mathbf{q}^{(1)}$ and $\mathbf{a}_j, j = 1, 2, \cdots, d$. The jth document is returned as relevant to the query only if $\cos\theta_j$ is greater than a certain threshold value. Typically a stringent cutoff value like

0.9 is used. In this simple example, we will use 0.5.

We find that

$$\cos\theta^{(1)} = [0.8165 \quad 0 \quad 0 \quad 0.5774 \quad 0],$$

and thus the first and the fourth books are returned.

Next if the user simply requested books about baking, then the query vector is

$$\mathbf{q}^{(2)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

and the resulting cosines are

$$\cos\theta^{(2)} = [0.5774 \quad 0 \quad 0 \quad 0.4082 \quad 0].$$

Thus only the first book is returned this time. The interesting consequence is that the fourth book, which is in fact a more comprehensive reference book about baking, is not returned as relevant!

Such failures can be alleviated by replacing $\mathbf{A}$ by a low-rank approximation to remove noise and uncertainties from the db representation.

4. The QR Factorization

The QR factorization can be used to identify and remove redundant information in the matrix representation of the db. Notice that $\hat{\mathbf{A}}$ does not have a rank of 5, rather its rank is 4 since column 5 is the sum of columns 2 and 3.

Even greater dependence can be expected in practice because

1. db of library materials can contain different editions of a book
2. db of Internet sites can contain a multitude of mirrors of the same web page
3. a db can easily have dependencies among it columns. For example, binary vectors representing documents "applied linear algebra" and "computer graphics" sum to the binary vector representing "linear algebra applied to computer graphics". Thus

a db containing all three documents would have dependencies among them its columns.

4.1. Identifying a Basis for the Column Space

The QR factorization of **A** is

$$\mathbf{A} = \mathbf{QR},$$

where **Q** is a $t \times t$ orthogonal matrix and **R** is a $t \times d$ upper triangular matrix. Since

$$A.j = \sum_{k=1}^{t} Q_{.k} R_{kj} = Q_{.1} R_{1j} + \cdots + Q_{.t} R_{tj},$$

and $Q_{.k}$ is the kth column of **Q**, therefore the jth column of **A** is a linear combination of the columns of **Q**. We also recall that the rank of **A**, r_A, is equal to the rank of **R**, which in turn is equal to the rank of $\mathbf{R}_A$ (=4 here), where $\mathbf{R}_A$ is the top nonzero part of **R**.

In our example, we have

$$\begin{aligned}\mathbf{Q} &= [\mathbf{Q}_A \mathbf{Q}_A^{\perp}] \\ &= \begin{bmatrix} -0.5774 & 0 & -0.4082 & 0 & 0.7071 & 0 \\ -0.5774 & 0 & 0.8165 & 0 & 0 & 0 \\ -0.5774 & 0 & -0.4082 & 0 & -0.7071 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & -0.7071 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & -0.7071 \end{bmatrix}\end{aligned}$$

and

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_A \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} -1 & 0 & -0.5774 & -0.7071 & -0.4082 \\ 0 & -1 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ 0 & 0 & 0 & -0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Therefore the economized representation

$$\mathbf{A} = [\mathbf{Q}_A \mathbf{Q}_A^{\perp}] \begin{bmatrix} \mathbf{R}_A \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_A \mathbf{R}_A.$$

The first r_A column of **Q** forms a basis for the columns of **A**.

In general it is important to use column pivoting to ensure that the zeros appear at the bottom of **R** and no zero appear on the diagonal of the upper part of **R**. The factorization gives **Q**, **R**, and **P** such that **AP** = **QR**, where **P** is a permutation matrix. **AP** amounts to a simple reordering of the document vectors. We will assume hereafter that this reordering has been performed and let **A** to represent **AP**.

In Matlab, **Q**, **R**, and **P** can be obtained by

```
[Q, R, P] = qr(A)
```

The semantic content of a db is fully described by any basis for the column space of **A**. Query matching now proceeds with **Q** and **R** in place of **A**.

Let $\mathbf{e}^{(j)}$ be a vector given by the jth column of the identity matrix, thus its nth element is given by δ_{jn}, where δ is the Knonecker symbol.

For any matrix, $\mathbf{C}$, $\mathbf{C}\mathbf{e}^{(j)}$ is the jth column of $\mathbf{C}$ because

$$(\mathbf{C}\mathbf{e}^{(j)})_m = \sum_n C_{mn}(\mathbf{e}^{(j)}) = \sum_n C_{mn}\delta_{jn} = C_{mj}.$$

Thus multiplying

$$\mathbf{A} = \mathbf{Q}_A\mathbf{R}_A.$$

on either side by $\mathbf{e}^{(j)}$, we see that

$$\mathbf{a}_j = \mathbf{Q}_A\mathbf{r}_j,$$

where $\mathbf{a}_j$ is the jth column of $\mathbf{A}$ and $\mathbf{r}_j$ is the jth column of $\mathbf{R}_A$.

Therefore we have

$$\cos\theta_j = \frac{\mathbf{a}_j^T\mathbf{q}}{\|\mathbf{a}_j\|_2\|\mathbf{q}\|_2} = \frac{(\mathbf{Q}_A\mathbf{r}_j)^T\mathbf{q}}{\|\mathbf{Q}_A\mathbf{r}_j\|_2\|\mathbf{q}\|_2} = \frac{(\mathbf{Q}_A\mathbf{r}_j)^T\mathbf{q}}{\|\mathbf{r}_j\|_2\|\mathbf{q}\|_2}.$$

4.2. Geometry of the Vector Space Model

Let $\mathcal{S}$ be the space spanned by the columns of $\mathbf{A}$, and let $\mathcal{S}^\perp$ its complementary space. So every vector in $\mathcal{S}$ is perpendicular to every vector in $\mathcal{S}^{\perp}$. Now if $\mathbf{I}$ is the $t \times t$ identity matrix, then

$$\begin{aligned} \mathbf{I} &= \mathbf{Q}\mathbf{Q}^T = [\mathbf{Q}_A \mathbf{Q}_A^{\perp}][\mathbf{Q}_A \mathbf{Q}_A^{\perp}]^T \\ &= [\mathbf{Q}_A \mathbf{Q}_A^{\perp}] \begin{bmatrix} \mathbf{Q}_A^T \\ (Q_A^{\perp})^T \end{bmatrix} = \mathbf{Q}_A \mathbf{Q}_A^{\perp} + Q_A^{\perp}(Q_A^{\perp})^T. \end{aligned}$$

Therefore the query vector $\mathbf{q}$ can be resolved into 2 mutually orthogonal components

$$\mathbf{q} = \mathbf{I}\,\mathbf{q} = \mathbf{Q}_A \mathbf{Q}_A^{\perp}\mathbf{q} + Q_A^{\perp}(Q_A^{\perp})^T\mathbf{q} = \mathbf{q}_A + \mathbf{q}_A^{\perp}.$$

Clearly $\mathbf{q}_A$ is the orthogonal projection of $\mathbf{q}$ into $\mathcal{S}$, and $\mathbf{q}_A^{\perp}$ is the orthogonal projection into $\mathcal{S}^{\perp}$.

Let $\mathbf{x}$ be an arbitrary vector in $\mathcal{S}$. Then using Pythagorean theorem, we have

$$\|\mathbf{q}-\mathbf{x}\|_2^2 = \|(\mathbf{q}-\mathbf{q}_A)+(\mathbf{q}_A-\mathbf{x})\|_2^2 = \|\mathbf{q}-\mathbf{q}_A\|_2^2+\|\mathbf{q}_A-\mathbf{x}\|_2^2 \geq \|\mathbf{q}-\mathbf{q}_A\|_2^2.$$

Thus

$$\|\mathbf{q}-\mathbf{q}_A\|_2^2 = \min_{\mathbf{x}\in\mathcal{S}} \|\mathbf{q}-\mathbf{x}\|_2^2.$$

That is $\mathbf{q}_A$ is the closest approximation of the query vector $\mathbf{q}$ in $\mathcal{S}$.

Next $\mathbf{a}_j$ must be perpendicular to $\mathbf{q}_A^\perp$ since they are in complementary spaces. More specifically we have

$$\mathbf{a}_j^T \mathbf{Q}_A^\perp = (\mathbf{Q}_A \mathbf{r}_j)^T \mathbf{Q}_A^\perp = \mathbf{r}_j \mathbf{Q}_A^T \mathbf{Q}_A^\perp = 0,$$

since any column in $\mathbf{Q}_A$ is perpendicular to any column in $\mathbf{Q}_A^\perp$. Therefore we have

$$\cos\theta_j = \frac{\mathbf{a}_j^T(\mathbf{q}_A + \mathbf{q}_A^\perp)}{\|\mathbf{a}_j\|_2 \|\mathbf{q}\|_2} = \frac{\mathbf{a}_j^T \mathbf{q}_A}{\|\mathbf{a}_j\|_2 \|\mathbf{q}\|_2}.$$

So only the $\mathbf{q}_A$ component of $\mathbf{q}$ contribute to the dot product used to compute $\cos\theta_j$.

Now if we replace $\mathbf{q}$ with $\mathbf{q}_A$, and compute a new measure of similarity by

$$\cos\tilde{\theta}_j = \frac{\mathbf{a}_j^T \mathbf{q}_A}{\|\mathbf{a}_j\|_2 \|\mathbf{q}_A\|_2},$$

which is clearly larger than $\cos\theta_j$, more relevant documents will be retrieved (*i*.e. a high recall), but at the expense of reducing the precision.

5. The Low-Rank Approximation

A term-by-document matrix $\mathbf{A}$ might be better represented by a matrix sum $\mathbf{A}+\mathbf{E}$, where the uncertainty matrix $\mathbf{E}$ may have any number of values reflecting missing or incomplete information about documents or even different opinions on the relevancy of documents to certain subjects.

$\mathbf{A}+\mathbf{E}$ may have a rank lower than that of $\mathbf{A}$. Lowering the rank may help to remove extraneous information or noise from the matrix representation of the db.

We need a notion of the size of a matrix We will use the Frobenius norm, which for any $t \times d$ matrix, $\mathbf{M}$ is defined as

$$\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^{t}\sum_{j=1}^{d} m_{ij}^2} = \sqrt{\sum_{i=1}^{t}\sum_{j=1}^{d} m_{ij}(\mathbf{M}^T)_{ji}} = \sqrt{\operatorname{tr}(\mathbf{M}\mathbf{M}^T)}.$$

In Matlab, the Frobenius norm of $\mathbf{M}$ is given by

```
norm(M,'fro')
```

We recall that or any square matrix, its trace is given by the sum of its diagonal elements. Also for any $m \times n$ matrix $\mathbf{S}$ and $n \times m$ matrix $\mathbf{T}$, we have $\mathrm{tr}(\mathbf{ST}) = \mathrm{tr}(\mathbf{TS})$.

Premultiplying a $t \times d$ matrix, $\mathbf{M}$, by a $t \times t$ orthogonal matrix $\mathbf{X}$ leaves the Frobenius norm unchanged:

$$\|\mathbf{XM}\|_F^2 = \mathrm{tr}((\mathbf{XM})^T\mathbf{XM}) = \mathrm{tr}((\mathbf{M}^T\mathbf{X}^T\mathbf{XM}) = \mathrm{tr}((\mathbf{M}^T\mathbf{M}) = \|\mathbf{M}\|_F^2.$$

Postmultiplying $\mathbf{M}$, by a $d \times d$ orthogonal matrix $\mathbf{Y}$ also leaves the Frobenius norm unchanged:

$$\begin{aligned} \|\mathbf{MY}\|_F^2 &= \mathrm{tr}((\mathbf{MY})^T\mathbf{MY}) = \mathrm{tr}((\mathbf{Y}^T\mathbf{M}^T\mathbf{MY}) = \mathrm{tr}((\mathbf{YY}^T\mathbf{M}^T\mathbf{M}) \\ &= \mathrm{tr}((\mathbf{M}^T\mathbf{M}) = \|\mathbf{M}\|_F^2. \end{aligned}$$

Recall that the rank of$\mathbf{A}$ is given by the number of nonzero diagonal elements in $\mathbf{R}$. These nonzero diagonal elements are ordered from large to small magnitudes by column pivoting in the QR decomposition.

We now consider partitioning $\mathbf{R}$ as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} = \left[\begin{array}{ccc|cc} -1 & 0 & -0.5774 & -0.7071 & -0.4082 \\ 0 & -1 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ \hline 0 & 0 & 0 & -0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

We find that

$$\frac{\|\mathbf{R}_{22}\|_F}{\|\mathbf{R}\|_F} = \frac{0.5774}{2.2361} = 0.2582,$$

and so $\mathbf{R}_{22}$ is a relatively small part of $\mathbf{R}$. If we remove $\mathbf{R}_{22}$ and define a new upper triangular matrix

$$\tilde{\mathbf{R}} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

which clearly has a lower rank (=3 for our example). Let us define

$\mathbf{AE} = \mathbf{Q}\tilde{\mathbf{R}}$, and so

$$\mathbf{E} = \mathbf{Q}\tilde{\mathbf{R}} - \mathbf{A} = \mathbf{Q}\tilde{\mathbf{R}} - \mathbf{QR} = \mathbf{Q}(\tilde{\mathbf{R}} - \mathbf{R}) = \mathbf{Q}\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -R_{22} \end{bmatrix}.$$

The square of its Frobenius norm is

$$\begin{aligned} \|\mathbf{E}\|_F^2 &= \left\| \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -R_{22} \end{bmatrix} \right\|_F^2 = \operatorname{tr}\left(\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -R_{22}^T \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -R_{22} \end{bmatrix} \right) \\ &= \|\mathbf{R}_{22}\|_F^2. \end{aligned}$$

Because

$$\|\mathbf{A}\|_F^2 = \|\mathbf{QR}\|_F^2 = \|\mathbf{R}\|_F^2,$$

we have

$$\frac{\|\mathbf{E}\|_F}{\|\mathbf{A}\|_F} = \frac{\|\mathbf{R}_2 2\|_F}{\|\mathbf{R}\|_F},$$

thus the relative change in **A** is the same as the relative change in **R**, which has a value of 0.2582 in our example. Uncertainty of this magnitude can be introduced simply by disagreement between expert indexers.

Replacing $\mathbf{A}$ by $\mathbf{A} + \mathbf{E}$ by using $\tilde{\mathbf{R}}$ and the first 3 columns of $\mathbf{Q}$ to compute the cosines gives

$$\cos\theta^{(1)} = [0.8165 \quad 0 \quad 0 \quad 0.7071 \quad 0]$$

for $\mathbf{q}^{(1)}$, thus returning the first and the fourth documents as before. The more interesting case is for $\mathbf{q}^{(2)}$ which now gives

$$\cos\theta^{(2)} = [0.5774 \quad 0 \quad 0 \quad 0.5 \quad 0].$$

Thus in addition to document 1, document 4, which we failed to return before, is now retrieved as relevant. We have improved the recall ability.

Next, if we push the rank reduction further, including now only the first 2 rows of $\mathbf{R}$ and the first 2 columns of $\mathbf{Q}$, we see that the relative change in $\mathbf{A}$ is rather large:

$$\frac{\|\mathbf{E}\|_F}{\|\mathbf{A}\|_F} = \frac{\|\mathbf{R}_{22}\|_F}{\|\mathbf{R}\|_F} = 0.5146.$$

The result for $\mathbf{q}^{(1)}$ is

$$\cos\theta^{(1)} = [0.8165 \quad 0 \quad 08165 \quad 0.7071 \quad 0.4083],$$

thus only only returning the first and the fourth books as before, the third book on Numerical Recipes is also retrieved! This book certainly have nothing to do with baking, and thus precision has been lowered.

We have precision problem also with $\mathbf{q}^{(2)}$, which now gives

$$\cos\theta^{(2)} = [0.5774 \quad 0 \quad 0.5774 \quad 0.5 \quad 0.2887].$$

Thus in both queries, irrelevant documents are incorrectly identified, and so we should not have pushed too far with rank reduction.

In general, the choice of the threshold cutoff value and the degree of rank reduction depend very much on the details of the particular db. A lot of fine-tuning and experimentation are needed to make IR really work well.

6. The Singular Value Decomposition

QR factorization gives a reduced-rank basis for the column space of $\mathbf{A}$, it gives no such information about its row space. Other than the documents themselves, the terms chosen for the db may also very well have noise and dependencies among them. The solution is to use singular value decomposition (SVD), which although is more expensive to compute, provides simultaneous reduced-rank approximation to both the column and the row spaces.

The SVD of the term-by-document matrix is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ is a $t \times t$ orthogonal matrix having the left singular vectors of $\mathbf{A}$ in its columns, $\mathbf{V}$ is a $d \times d$ orthogonal matrix having the right singular vectors of $\mathbf{A}$ in its columns, and $\mathbf{\Sigma}$ is a $t \times d$ diagonal matrix having singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(t,d)} \geq 0$ of $\mathbf{A}$ along its diagonal. This factorization exist for any matrix $\mathbf{A}$. Recall also that the rank of $\mathbf{A}$, r_A, is given by the number of nonzero singular values. The first r_A columns of $\mathbf{U}$ form a basis for the column space of $\mathbf{A}$, and the first r_A rows of $\mathbf{V}^T$ form a basis for the row space of $\mathbf{A}$. We can create a rank-k approximation to $\mathbf{A}$, $\mathbf{A}^{(k)}$, by setting all but the k largest singular values of $\mathbf{A}$ to zero.

The theorem by Eckart and Young implies that the distance between $\mathbf{A}$ and any of its rank-k approximants is minimized by $\mathbf{A}^{(k)}$.

Moreover on has

$$\min_{\operatorname{rank}(\mathbf{X})\leq k} \|\mathbf{A}-\mathbf{X}\|_F = \|\mathbf{A}-\mathbf{A}|_F = \sqrt{(\sigma_{k+1}^2+\cdots+\sigma_{r_A}^2)}.$$

The Frobenius norm of $\mathbf{A}$ is simply related to the singular values:

$$\begin{aligned} \|\mathbf{A}\|_F &= \|\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\|_F = \|\boldsymbol{\Sigma}\mathbf{V}^T\|_F = \|\mathbf{V}\boldsymbol{\Sigma}^T\|_F = \|\boldsymbol{\Sigma}^T\|_F \\ &= \sqrt{\sum_{j=1}^{r_A}\sigma_j^2}. \end{aligned}$$

The rank-k approximation to $\mathbf{A}$ is

$$\mathbf{A}^{(k)} = \mathbf{U}^{(k)}\boldsymbol{\Sigma}^{(k)}(\mathbf{V}^{(k)})^T,$$

where $\mathbf{U}^{(k)}$ is the $t\times k$ matrix whose columns are the first k columns of $\mathbf{U}$, $\mathbf{V}^{(k)}$ is the $d\times k$ matrix whose columns are the first k columns of $\mathbf{V}$, and $\boldsymbol{\Sigma}^{(k)}$ is the $k\times k$ diagonal matrix whose diagonal elements are $\sigma_1,\cdots\sigma_k$, the k largest singular value of $\mathbf{A}$.

From the Eckart and Young theorem we have $\|\mathbf{A}-\mathbf{A}^{(3)}\|_F = \sigma_4 = 0.4195$, and so $\|\mathbf{A}-\mathbf{A}^{(3)}\|_F/\|\mathbf{A}\|_F = 0.4195/2.2361 = 0.1876$.

On the other hand, $\|\mathbf{A} - \mathbf{A}^{(2)}\|_F = \sqrt{\sigma_3^2 + \sigma_4^2} = 0.9392$, and so $\|\mathbf{A}-\mathbf{A}^{(2)}\|_F/\|\mathbf{A}\|_F = 0.9392/2.2361 = 0.4200$. These relative changes are much smaller than the corresponding changes, 0.2582 and 0.5146, using the QR method. Therefore we expect $\mathbf{A}^{(3)}$ to represent the structure of the db well.

It is also interesting to note that the rank-3 approximation to $\mathbf{A}$ in the QR method looks much closer to the corresponding matrix in the SVD method. The latter contains negative elements resulting from linear combination from the columns of $\mathbf{A}$. Interpretation of the elements as related to the number of times a certain term appear in a given document no longer applies. However that should not present any problem, since the db content is modeled by the geometric relationship between the columns of $\mathbf{A}$ and the query vectors.

7. The Reduced-Rank Vector Space Model

We now compare the query vector $\mathbf{q}$ to the columns of $\mathbf{A}^{(k)}$. By defining $\mathbf{e}^{(j)}$ to be a vector of length d such that $(\mathbf{e}^j)_i = \delta_{ij}$, we can write

$$\begin{aligned}\cos\theta_j &= \frac{(\mathbf{A}^{(k)}\mathbf{e}^{(j)})^T\mathbf{q}}{\|\mathbf{A}^{(k)}\mathbf{e}^{(j)}\|_2\|\mathbf{q}\|_2} = \frac{(\mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{V}^{(k)})^T\mathbf{e}^{(j)})^T\mathbf{q}}{\|\mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{V}^{(k)})^T\mathbf{e}^{(j)}\|_2\|\mathbf{q}\|_2}\\ &= \frac{(\mathbf{e}^{(j)})^T\mathbf{V}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{U}^{(k)})^T\mathbf{q}}{\|\mathbf{\Sigma}^{(k)}(\mathbf{V}^{(k)})^T\mathbf{e}^{(j)}\|_2\|\mathbf{q}\|_2}.\end{aligned}$$

The above equation can be rewritten in terms of the following vector of length k

$$\mathbf{s}^{(j)} = \mathbf{\Sigma}^{(k)}(\mathbf{V}^{(k)})^T\mathbf{e}^{(j)},$$

as

$$\cos\theta_j = \frac{(\mathbf{s}^{(j)})^T(\mathbf{U}^{(k)})^T\mathbf{q}}{\|\mathbf{s}^{(j)}\|_2\|\mathbf{q}\|_2}.$$

It is clear that $(\mathbf{U}^{(k)})^T\mathbf{q}$ is the projection of $\mathbf{q}$ into the column space of $\mathbf{A}^{(k)}$. As in the QR factorization, we introduce a larger cosine term

$$\cos\tilde{\theta}_j = \frac{(\mathbf{s}^{(j)})^T(\mathbf{U}^{(k)})^T\mathbf{q}}{\|\mathbf{s}^{(j)}\|_2\|(\mathbf{U}^{(k)})^T\mathbf{q}\|_2}.$$

to improve recall (of course at the possible expense of precision).

We now revisit the query $\mathbf{q}^{(1)}$ for books about baking bread. Using now the rank-3 approximation, $\mathbf{A}^{(3)}$, we find that

$$\cos\tilde{\theta}^{(1)} = [0.7327 \quad -0.0469 \quad 0.0330 \quad 0.7161 \quad -0.0097].$$

The first and the fourth books are still correctly identified, with no irrelevant books incorrectly returned. For the query $\mathbf{q}^{(2)}$ about baking bread, $\mathbf{A}^{(3)}$ gives

$$\cos\tilde{\theta}^{(2)} = [0.5181 \quad -0.0332 \quad 0.0233 \quad 0.5064 \quad -0.0069].$$

Both books about baking are returned with similar ratings, and no irrelevant books are incorrectly returned.

However if we push one step further and use the rank-2 approximation, $\mathbf{A}^{(2)}$, we find that

$$\cos\tilde{\theta}^{(1)} = [0.5181 \quad -0.1107 \quad 0.5038 \quad 0.3940 \quad 0.2362].$$

The first book is correctly returned. But the Numerical Recipes book has a rating almost as high as the first book. It is incorrectly returned. The fourth book, which is perhaps the best match, does not even make the cut, and will not be retrieved.

For the query $\mathbf{q}^{(2)}$ about baking bread, $\mathbf{A}^{(2)}$ gives

$$\cos\tilde{\theta}^{(2)} = [0.3663 \quad -0.0783 \quad 0.3563 \quad 0.2786 \quad 0.1670].$$

whose values are all less than 0.5. Thus not even a single book is retrieved. Clearly $\mathbf{A}^{(2)}$ is not a reasonable approximation to $\mathbf{A}$.

8. Term-Term Comparison

Up to this point, we have been concerned with the vector space model as a mechanism for comparing queries with documents. With minor variation, the model can also be used to compare terms with terms. When implemented as part of a search engine, a term-term comparison helps to refine the results of a search automatically.

9. Other Techniques that Really make IR Work

9.1. Relevance Feedback

Due to polysemy and synonymy, a list of document retrieved for for a given query is never perfect; user has to ignore some of the returned items.

Precision can be improved using relevance feedback, where the user helps to specify which document from a returned set are most relevant. That information is then used to clarify the intend of the original query.

Term-term comparison can be used for relevance feedback to improve a query based on term clustering information.

Relevance feedback can also be carried out in the column space of the $t \times d$ matrix. The query is supplemented or even replaced by the vector sum of the most relevant document returned in order to focus the search nearer those document vectors.

9.2. Managing Dynamic Collections

Information is constantly added or removed, meaning that catalogs and indexes become obsolete or incomplete (sometimes in a matter of seconds). The most obvious approach to accommodating additions (new terms or documents) is to recompute the SVD of the new term-by-document matrix, but, for large databases, this procedure is very costly in time and space. Less expensive alternatives, such as folding-in and SVD-updating, have been examined.

9.3. Downdating

Using SVD-downdating, the present model can be modified to reflect the removal of terms and documents and any subsequent changes to term weights. Downdating can be useful for information filtering (e.g., parental screening of Internet sites) and evaluating the importance of a term or document with respect to forming or breaking clusters of semantically related information.

9.4. Sparsity

The sparsity of a term-by-document matrix is a function of the word usage patterns and topic domain associated with the document collection. The more new terms each document brings to the global dictionary, the sparser is the matrix overall. The sample IR matrices studied are typically no more than 1% dense. Exploitation of the sparcity can significantly improve response times, updating and downdating processing times.

References

[1] W. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1992. 3

[2] G. Kowalski, *Information Retrieval Systems: Theory and Implementation*, Kluwer Academic Publishers, Boston, 1997. 3

[3] Material here is taken from the article by Michael W. Berry, Zlatko Drmač, and Elizabeth R. Jessup, *Matrices, Vector Spaces, and Information Retrieval*, SIAM REVIEW, Society for Industrial and Applied Mathematics, Vol. **41**, No. 2, pp. 335362, 1999.

3