Matrices, Vector Spaces, and Information Retrieval

Authors: M. W. Berry and Z. Drmac and E. R. Jessup
SIAM 1999: Society for Industrial and Applied Mathematics
Speaker: Mattia Parigiani
Introduction

- Large volumes of data need to be handled with advent of digital libraries & the Internet
- Classical methods can’t handle it
- Recent IR techniques make use of linear algebra, the VSM
- Data modeled as a matrix & user query as a vector
- Vector operations used to retrieve relevant documents
- Matrix factorizations used to handle uncertainties & noise in the database
Problem

Today:

- 60,000 books printed every year in the USA
- 300,000,000 web pages on Internet. Search engines acquire about 10 million pointers daily

⇒ Need efficient automated IR techniques
⇒ Need for better indexing strategies
Authors show how linear algebra and the VSM can be used for automated IR

Documents encoded as vectors where elements represent:

- A term for the document
- The weight of the term in representing the document

Document collection = A matrix

A user query = set of terms in a vector (similar format as a document vector)

Finally: vector operations used to find relevant documents in the DB representation
Main Techniques: Overview

- **QR FACTORIZATION**
  - A type of orthogonal factorization
  - To show basics of Rank-Reduction
  - To provide geometric interpretation of the VSM
  - To account for uncertainties in the DB

- **SVD – Singular Value Decomposition**
  - Stronger orthogonal factorization method
  - Allows comparisons between:
    - Document & Document
    - Term & Term
Example – VSM

- “applied linear algebra” – terms used to index a document
- Suppose weights are: 0.5, 2.5, 5.0
- “algebra” = most important term
- Geometric representation:
VSM Representation

- If we have \( d \) documents & \( t \) terms a DB is represented by a \((t \times d)\) term-by-document matrix \( A \):
  - Columns : Document Vectors
  - Rows : Term Vectors
  - \( A(i,j) = \) weighted frequency of the \( i^{th} \) term associated with the \( j^{th} \) document
Wrap-up Example

The $t = 6$ terms:  

| T1: | bak(e,ing) |
| T2: | recipes    |
| T3: | bread      |
| T4: | cake       |
| T5: | pastr(y,ies) |
| T6: | pie        |

The $d = 5$ document titles:

| D1:  | How to Bake Bread Without Recipes |
| D2:  | The Classic Art of Viennese Pastry |
| D4:  | Breads, Pastries, Pies and Cakes: Quantity Baking Recipes |
| D5:  | Pastry: A Book of Best French Recipes |

The $6 \times 5$ term-by-document matrix before normalization, where the element $\hat{a}_{ij}$ is the number of times term $i$ appears in document title $j$:

$$
\hat{A} = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 
\end{pmatrix}
$$

The $6 \times 5$ term-by-document matrix with unit columns:

$$
A = \begin{pmatrix}
0.5774 & 0 & 0 & 0.4082 & 0 \\
0.5774 & 0 & 1.0000 & 0.4082 & 0.7071 \\
0.5774 & 0 & 0 & 0.4082 & 0 \\
0 & 0 & 0 & 0.4082 & 0 \\
0 & 1.0000 & 0 & 0.4082 & 0.7071 \\
0 & 0 & 0 & 0.4082 & 0 
\end{pmatrix}
$$
Query Evaluation

- **Query Matching** – Finding most geometrically close vectors in the VSM matrix to the query vector
- Usually calculates the *cosine* of the angle between the vectors

\[ \cos \theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^{t} a_{ij} q_i}{\sqrt{\sum_{i=1}^{t} a_{ij}^2} \sqrt{\sum_{i=1}^{t} q_i^2}} \]

- \( \cos(0) = 1 \), hence parallel vectors, hence similar documents
Example Continued – Query Matching

- Suppose query “baking bread”
- Query Vector $\mathbf{q} = [1, 0, 1, 0, 0, 0]^T$
- If cosine between $\mathbf{q}$ and single document vector $> \text{threshold}$
  $\Rightarrow$ Relevant document found!!
- Small example – $\text{threshold} = 0.5$
- 1st and 4th documents retrieved
Example Continued – Problems & Conclusion

- But if vector becomes “baking”
- \( q = [1,0,0,0,0,0]^T \)
- Only first document retrieved!! And the 4th?
- IR community has worked-around these type of failures
  - Controlled Vocabulary
  - Rank-Reduction of document Matrix (LSI basis)
  - QR-Factorization
  - Singular Value Decomposition (SVD)
QR Factorization

- Remove redundant information from the VSM matrix
- Simplifies cost of query evaluation
- Method identifies dependencies between columns/rows in the term-by-document matrix
- The factorization has the form:
  - \( A = QR \)
  - \( R - (t \times d) \) upper triangular matrix
  - \( Q - (t \times t) \) orthogonal matrix
- Different methods are available for computing the QR Factorization
Example – Continued

- From $A = QR$, the factors are:

$$Q = \begin{pmatrix} -0.5774 & 0 & -0.4082 & 0 & -0.7071 & 0 \\ -0.5774 & 0 & 0.8165 & 0 & 0.0000 & 0 \\ -0.5774 & 0 & -0.4082 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & -0.7071 \\ 0 & -1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.7071 & 0 & 0.7071 \end{pmatrix}$$

$$R = \begin{pmatrix} -1.0001 & 0 & -0.5774 & -0.7070 & -0.4082 \\ 0 & -1.0000 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ 0 & 0 & 0 & -0.5774 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Partitioning used for algebraic simplifications
QR Factorization: Conclusion

- Query matching can now be done more efficiently on the factors $\textbf{QR}$ other than on $\textbf{A}$.
- The new cosine formula looks like:

$$
\cos \theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{(Q_A r_j)^T q}{\|Q_A r_j\|_2 \|q\|_2} = \frac{r_j^T (Q_A^T q)}{\|r_j\|_2 \|q\|_2}
$$

- There is no loss of information in using this factorized form.
The Low-Rank Approximation

- QR Factorization mainly deals with uncertainties in the database.
- Indexing process can lead to uncertainties in the matrix: built by people with different experiences & opinions.
- Better to use a matrix $A + E$:
  - $E = \text{uncertainty matrix}$
  - Missing/incomplete information; relevancy of document to certain subjects; other opinions.
The Low-Rank Approximation – 2

- Finds a matrix $E$ such that $(A+E)$ has a smaller rank than $A$.
- Lowering the rank of $A$ helps to remove extraneous information & noise from the matrix representation.
- This is demonstrated using properties & application of the Frobenius Norm.
- $A=QR$ is used in this process.
The Low-Rank Approximation – 3

- **A=QR** used to manipulate $r_R$ by separating the small parts of $R$ from the big parts.

- From example, $R = \begin{pmatrix} -1.0001 & 0 & -0.5774 & -0.7071 & -0.4082 \\ 0 & -1.0000 & 0 & -0.4082 & -0.7071 \\ 0 & 0 & 0.8165 & 0 & 0.5774 \\ 0 & 0 & 0 & -0.5771 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = (R_{11} R_{12})$.

- A small part can be isolated, $R_{22}$, then set to ZERO => new matrix $R'$

- $R'$ has rank = 3

- Also $A+E = QR'$ has rank = 3

- $E$ given by difference:

  $$E = (A + E) - A = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} - Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = Q \begin{pmatrix} 0 & 0 \\ 0 & -R_{22} \end{pmatrix}$$
The Low-Rank Approximation – 4

- Using Frobenius theorem & norm:
  - 26% change in the value of $\mathbf{R}$ gives equal 26% change in the value of $\mathbf{A}$
  - Change reduces the rank of both matrices by 1!
- 26% is same order of disagreement introduced by 2 indexers (study case)
  - Hence, rank-3 approximation $\mathbf{A} + \mathbf{E}$ can be accepted as good to replace $\mathbf{A}$
Matrix, Vector Spaces, and Information Retrieval

The Low-Rank Approximation – 5

- cosines are computed using the QR factor
- No need to calculate $A+E$
- Reducing to rank-2, we get a 52% relative change
  - Leads to retrieving non-relevant documents
  - Unacceptably large change

Conclusions:
- Query Matching improves if the rank of the VSM model is reduced
Singular Value Decomposition (SVD)

- QR Factorization gives a rank reduced basis for the column space of the term-by-document matrix.
- No information about the row space => no mechanism for term-to-term comparison.
- SVD expensive but gives a reduced rank approximation to both spaces.
- SVD can find a rank $k$ approximation of the term-by-document matrix with minimal change to the matrix.
Singular Value Decomposition (SVD) – 2

- The decomposition is: $A = U\Sigma V^T$
  - $U: (t \times t)$ orthogonal having the left singular vectors of $A$ as its columns
  - $V: (d \times d)$ orthogonal having the right singular vectors of $A$ as its columns
  - $\Sigma: (t \times d)$ diagonal having singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{\min(t,d)}$ of $A$ along its diagonal

- Factorization exists for any matrix $A$
- There are existing methods of elaborating the SVD
Singular Value Decomposition (SVD) – 3

- Many parallels between SVD and QR
  - The rank $r_A$ of $A = \#$ of nonzero diagonal elements of $R$, also = $\#$ number of nonzero elements of $\Sigma$
  - The first $r_A$ columns of $Q$ are a basis for the column space of $A$, the first $r_A$ columns of $U$ form the same basis

- Extra: Can identify a basis of the row space of $A$ in factor $V^T$

- Difference between the 2 based upon theoretical issues (Eckart & Young’s theorem)
Singular Value Decomposition (SVD) – 4

- Unlike the QR Factorization, SVD provides us with a lower rank representation of the column & row spaces.

- We know $A_k$ is the best rank-k approximation to $A$ by Eckert & Young’s Theorem:

$$\|A - A_k\|_F = \min_{\text{rank}(X) \leq k} \|A - X\|_F$$
Conclusions

- Using Eckert & Young’s theorem on the current example:
  - $A$ has only a 19% relative change when the rank goes from 4 to 3
  - The change is 42% if rank goes down to 2
- With $k = 3$, the resulting matrix $A_3$ spans a 3-dimensional subspace of the column space of $A$ which represents well the structure of the DB
- Percentage changes are better than in QR Factorization
Using the SVD for Query Matching

- **cosine** formula for the angles between the query & the columns of our rank-\(k\) approximation of \(A\)

\[
\cos \theta_j = \frac{s_j^T (U_k^T q)}{\|s_j\|_2 \|q\|_2}, \quad j = 1, \ldots, d,
\]

- Using the rank-3 approximation we return the 1\(^{st}\) & 4\(^{th}\) books again using threshold of 0.5
Term–Term Comparisons

- SVD can be used not only for query to document comparison but also for term to term comparison
- This can help refine the search results obtained from an initial query
Term–Term Comparisons – Example

\[ G \] represents the documents returned from the initial search using a **polisemous** key ‘run’

Compare all term vectors by

\[
\cos \omega_{ij} = \frac{(e_i^T G)(G^T e_j)}{\|G^T e_i\|_2 \|G^T e_j\|_2}
\]

\[ C(i,j) \] – how relevant term i is to term j
Clustering

- Clustering is the process by which terms are grouped if they are related such as *bike*, *endurance* and *training*
- Other strategies for clustering exist based on Graph Theory
IR Improvements – Relevance Feedback

- Ideal IR systems should achieve high precision without returning irrelevant documents.
- Not the case due to problems of polisemy & synonymy.
- *Relevance Feedback* – from given result list, user specifies which documents are more relevant to clarify meaning of original search.
- Term-Term comparison already seen is a first method.
- A 2nd method is based on the column space of the term-by-document matrix.
IR Improvements – Managing Dynamic Collections

- Database rarely stay the same: information constantly added & removed
- Hence catalogs and indexes become obsolete and incomplete
- A recomputation of SVD is required to get a new term-by-document matrix
- Can be costly for large DBs (time & space)
- Less expensive approaches exist:
  - Folding-In
  - SVD-Updating
Managing Dynamic Collections – Folding-In

Folding a new document vector into the column space of an existing term-by-document matrix amounts to finding coordinates for that document in the basis $U_k$.
Managing Dynamic Collections – SVD-Updating

- Accounts for maintaining orthogonality when new documents & terms are introduced

- 3 main steps:
  - updating terms
  - updating documents
  - updating term weights
Sparsity

- Sparsity of the term-by-document matrix is a function of the word usage patterns & topic domain associated with the document collection.
- Usually sparse matrices without patterns.
- Special matrix formats have been developed to compute the SVD:
  - *banded*
  - *enveloped*
- Special methods for computing the SVD on these matrices are also available.
Conclusion

- Through the use of this model many libraries and smaller collections can index their documents
- This methodologies and variants can be applied to the retrieval part in Case Base Reasoning systems