Matrix Factorizations for Information Retrieval

Dianne P. O'Leary

Computer Science Dept. and
Institute for Advanced Computer Studies
University of Maryland
oleary@cs.umd.edu

http://www.cs.umd.edu/users/oleary

Support: NSF, NIST, CCS

The Plan

- A catalog of matrix factorizations and their uses
  - Eigendecomposition
  - Pivoted QR
  - SVD
  - URV
  - SDD
  - non-negative factorizations
- An application: Building a retrieval system that is predominantly linear algebra
- Conclusions

A Catalog of Matrix Factorizations

The Eigendecomposition

The eigendecomposition of a matrix $\mathbf{A}$ of dimension $n \times n$ is

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with entries λ_i the eigenvalues. The columns of $\mathbf{U}$ are the right eigenvectors:

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \,,$$

and the rows of $\mathbf{U}^{-1}$ are the left eigenvectors $\mathbf{z}_i^T$, satisfying

$$\mathbf{z}_i^T \mathbf{A} = \lambda_i \mathbf{z}_i^T.$$

The decomposition is guaranteed to exist if

- $\mathbf{A}$ is real symmetric or complex Hermitian, or
- the eigenvalues of $\mathbf{A}$ are distinct. Otherwise, the decomposition may fail to exist, although it will exist for a nearby matrix.

Cost of the Eigendecomposition

- Cost: a large multiple of n^3 to compute all n eigenvalues and eigenvectors.
- For large problems, we might be interested in only a few of the eigenvalues and eigenvectors.
- In that case, we might use an iterative method in the Krylov subspace class of methods (e.g., Arnoldi, Lanczos), with cost per iteration proportional to the cost of multiplying a vector by $\mathbf{A}$.
- Updating the eigenvalues and eigenvectors using Arnoldi or Lanczos is inexpensive if the matrix entries are changed a bit.
- Updating the eigenvalues and eigenvectors is more expensive if the size of the matrix is changed.

Uses of the Eigendecomposition

- stability analysis
- structural analysis, testing for cracks
- designing to damp noise or oscillation
- understanding Markov chains

Example Use in Information Retrieval: PageRank

PageRank (Page, Brin 1998)

One variant: Let G be the Web graph:

- nodes = pages
- edge from page i to page j if i links to j

Now imagine a surfer who, starting from a given page, clicks randomly, with equal probability, among the links on that page, or, with very tiny probability, to some other page.

The PageRank of page i is the probability that the surfer will be visiting page i at some fixed time, after a large number of clicks.

- The PageRanks are the entries of the stationary vector of the Markov chain defined by the random walk.
- This vector is the eigenvector of the probability matrix corresponding to the eigenvalue that equals 1.

Structure to be Exploited in PageRank:

The size of the matrix is enormous, but

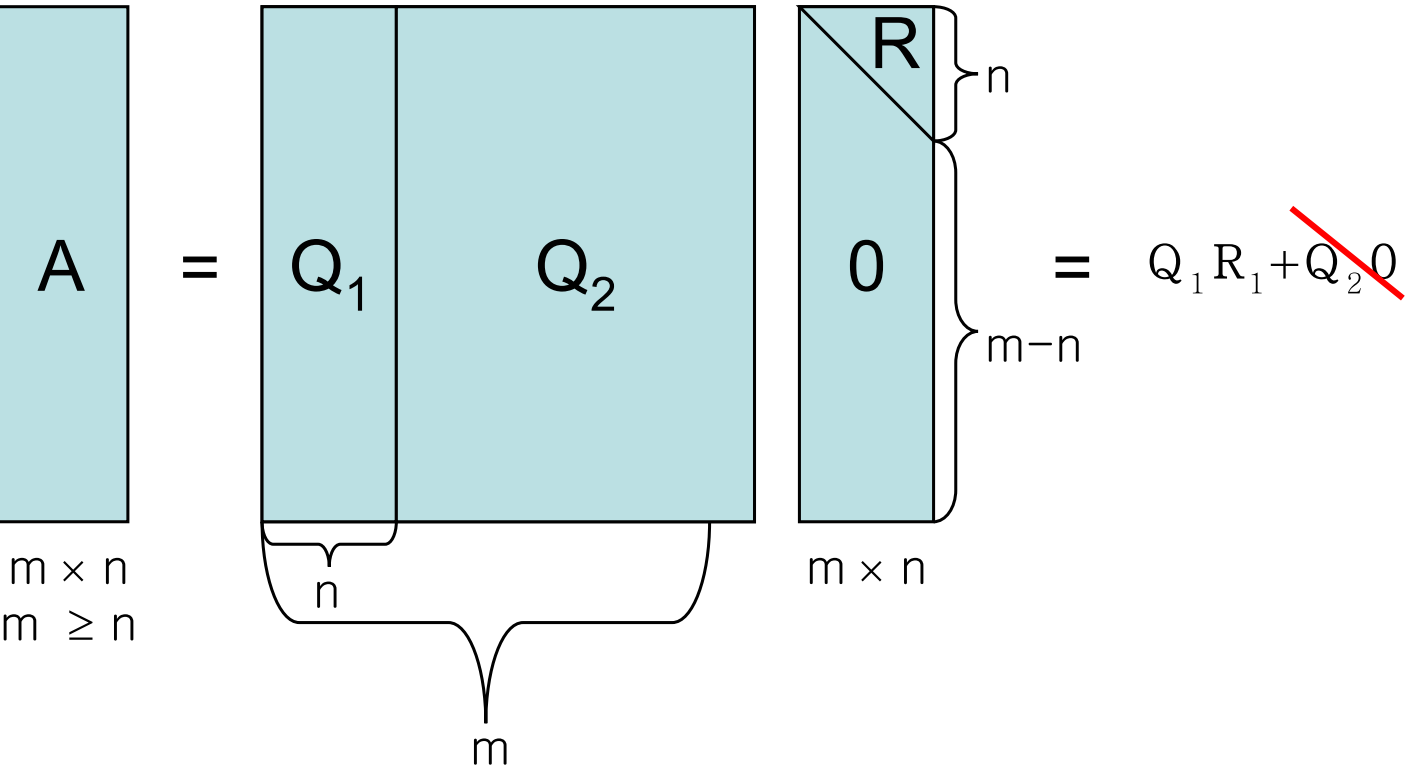
- It is a sparse piece plus a piece of small norm.
- We only need one eigenvector, not all of them.
- We can update an approximate solution when the link structure changes (Page, Brin), and Arnoldi-type methods are a natural alternative to the simplest update algorithm (Golub, Greif).

- The matrix has a great deal of block structure, and this can be used to accelerate the update iteration and induce parallelism. (Kamvar, Haveliwala, Manning, Golub)
- The bottleneck is often in data movement (Chen, Gan, Suel).
- Lumping and aggregation can be used to speed up the computation (Lee, Golub, Zenios; Langville, Meyer), and multilevel computations can be used (Bradley, de Jager, Knottenbelt, Trifunovic).

Rank-revealing QR Decomposition

Pivoted QR Decomposition

The QR decomposition of an $m \times n$ matrix $\mathbf{A}$ $(m \geq n)$ is defined by



$\mathbf{Q}$ is $m \times n$ and orthogonal, and $\mathbf{R}$ is $n \times n$ and upper triangular.

Rank-Revealing QR Decomposition

The RR-QR decomposition of an $m \times n$ matrix $\mathbf{A}$ is defined by $\mathbf{AP} = \mathbf{QR}$ where $\mathbf{P}$ is a permutation matrix chosen so that the leading principal submatrix of $\mathbf{R}$ of dimension $p \times p$ is well conditioned and the other diagonal block (of dimension $(n-p) \times (n-p)$) is small. The numerical rank of $\mathbf{A}$ is p.

Cost of the RR-QR Decomposition

Cost:

- The work is $O(mn^2)$.
- If only p columns of $\mathbf{A}$ are of interest, or if we want a rank-p approximation to $\mathbf{A}$, this can be reduced to $O(mnp)$.
- If p is small enough, this is relatively inexpensive. But see Pete Stewart's talk, next, for a sparse variant.
- Updating can be done rather inexpensively when elements are changed or rows/columns are added.

Uses of the RR-QR Decomposition

Uses:

- solving least squares problems

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$$

- finding orthogonal bases for the range and null-space of $\mathbf{A}$.
- estimating the rank of $\mathbf{A}$ (although SVD is more reliable).

Example Use in Information Retrieval: Redundancy Removal

Conroy, O'Leary, 2002

Suppose we have a term-sentence matrix $\mathbf{B}$, with the columns scaled to measure the importance of the corresponding sentence.

Suppose we want to construct a summary of the document represented by the matrix.

We can use pivoted-QR to do this:

- We first choose the sentence having maximum norm.

- Then, within the matrix $\mathbf{B}$, we subtract from each remaining sentence the component in the direction of this chosen sentence, and then we again choose the one with maximum norm.
- This process is iterated until the collection of chosen sentences is of the desired size.

The Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD)

Every matrix $\mathbf{A}$ of dimensions $m \times n$ ($m \geq n$) can be decomposed as

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

where

- $\mathbf{U}$ has dimension $m \times m$ its columns are orthogonal.
- $\mathbf{\Sigma}$ has dimension $m \times n$, the only nonzeros are on the main diagonal, and they are nonnegative real numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$,
- $\mathbf{V}$ has dimension $n \times n$ and its columns are orthogonal.

This is the SVD.

Properties of the SVD

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

or

$$\mathbf{A} \approx \mathbf{U}_p\mathbf{\Sigma}_p\mathbf{V}_p^T$$

where

- $\mathbf{U}_p$ is $m \times p$ with orthogonal columns,
- $\mathbf{\Sigma}_p$ is $p \times p$ and diagonal,
- $\mathbf{V}_p$ is $n \times p$ with orthogonal columns.

- The singular values σ_i of $\mathbf{A}$ are the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$.
- The columns of $\mathbf{V}$ are the right singular vectors of $\mathbf{A}$ and the eigenvectors of $\mathbf{A}^T\mathbf{A}$.

- The columns of $\mathbf{U}$, which are the left singular vectors of $\mathbf{A}$, are the eigenvectors of $\mathbf{A}\mathbf{A}^T$.

Cost of the SVD

- Cost: $O(mn^2)$. The constant is usually of order 10 and the first step is reducing the matrix to bidiagonal form.
- For large sparse matrices, the SVD can be computed iteratively through a bidiagonalization algorithm related to the Lanczos process (Golub et al.).
- Updating the SVD is not practical, but there are ways to update the bidiagonalization.

Uses of the SVD

This is the Swiss Army knife of matrix decompositions.

- solving ill-conditioned least squares problems
- solving discretized ill-posed problems
- solving linear systems
- determining the rank of a matrix
- determining a low-rank approximation to a matrix
- ...

Example Use in Information Retrieval: LSI

Deerwester, Dumais, Furnas, Landauer, Harshman, 1990

The SVD was the original factorization proposed for Latent Semantic Indexing (LSI), the process of replacing a term-document matrix $\mathbf{A}$ with a low-rank approximation $\mathbf{A}_p$ which reveals implicit relationships among documents that don't necessarily share common terms.

Example:

Term	D1	D2	D3	D4	D5
twain	53	65	0	30	1
clemens	10	20	40	43	0
huckleberry	30	10	25	52	70

- A query on clemens will retrieve D1, D2, D3, and D4.
- A query on twain will retrieve D1, D2, and D4.

For $p = 2$, the SVD gives

Term	D1	D2	D3	D4	D5
twain	49	65	7	34	-5
clemens	23	22	14	30	21
huckleberry	25	9	34	57	63

- Now a query on clemens will retrieve all documents.
- A query on twain will retrieve D1, D2, D4, and possibly D3.
- The negative entry is disturbing to some and motivates the nonnegative factorizations.

Necessary Properties for LSI

- significant compression
- rapid query evaluation
- ability to add new terms and documents
- recognition of latent relationships

SVD is not so easy to update, so we consider three alternatives: URV, SDD, and nonnegative factorizations.

Rank-revealing URV

Rank-revealing URV

Stewart 1990

Express $\mathbf{A}$ in the form

$$\mathbf{A} = \mathbf{URV}^T = \mathbf{U} \begin{bmatrix} \tilde{\mathbf{R}} & \mathbf{F} \\ \mathbf{0} & \mathbf{G} \end{bmatrix} \mathbf{V}^T$$

where

- the columns of $\mathbf{U}$ and $\mathbf{V}$ are orthogonal,
- $\tilde{\mathbf{R}}$ and $\mathbf{G}$ are upper triangular of orders p and $m-p$,
- $\mathbf{F}$ and $\mathbf{G}$ are small in norm.

This decomposition reveals that $\mathbf{A}$ is within $\sqrt{\|\mathbf{F}\|^2 + \|\mathbf{G}\|^2}$ of the matrix $\hat{\mathbf{A}}$ of rank p obtained by setting $\mathbf{F}$ and $\mathbf{G}$ to zero.

Cost of the Rank-revealing URV

- a modest multiple of mn^2.
- Updating is possible in $O(m^2)$ time (and in $O(m)$ time on a linear array of m processors).

Updating a URV Decomposition

The updating procedure consists of two parts:

- The incorporation step is analogous to the standard update of a QR decomposition but special care is taken that only the first column of F and G increases in norm.

  This corresponds to the fact that the addition of a row/column to a matrix can increase its rank by at most one.
- After the update, a condition estimator is used to test $\tilde{R}$ for rank degeneracy, and a deflation step reduces the norm of the last column of $\tilde{R}$. If a degeneracy is detected, a refinement step is performed to bring the decomposition closer to diagonal form.

Uses of the Rank-revealing URV

- The $\mathbf{U}$ and $\mathbf{V}$ of the URV decomposition are different than those of the SVD, but still provide approximate bases (exact in the absence of noise) for the required spaces.
- URV provides a less expensive alternative to SVD in LSI.

The Semidiscrete Decomposition (SDD)

The Semidiscrete Decomposition (SDD)

The Semi-Discrete Decomposition (SDD)
(O'Leary, Peleg 1983; Kolda, O'Leary 1998, 2000; O'Leary, Roth 2006)

SVD:

$$\tilde{\mathbf{A}}_{SVD} = \mathbf{U}_p \mathbf{\Sigma}_p \mathbf{V}_p^T$$

SDD:

$$\tilde{\mathbf{A}}_{SDD} = \mathbf{X}_p \mathbf{D}_p \mathbf{Y}_p^T$$

- $\mathbf{D}_p$ is diagonal
- The entries of $\mathbf{X}_p$ and $\mathbf{Y}_p$ are 0, 1, or -1.

Cost of the SDD

	SDD	SVD
Storage (bytes)	$4p + \frac{1}{4}p(m+n)$	$8p(m+n+1)$
Query Scoring	$p(m+n)$ adds, p mults	$p(m+n)$ adds, $p(1+m+n)$ mults

Updating algorithms have been proposed.

Uses of the SDD

- image compression
- LSI

Non-negative Factorizations

Non-negative Factorizations

Lee, Seung, 2001

A common theme: Given a matrix $\mathbf{A}$ of size $m \times n$, find $\mathbf{X}$ of size $m \times p$, a diagonal matrix $\mathbf{D}$, and $\mathbf{Y}$ of size $p \times n$ to solve:

$$\min_{\mathbf{X},\mathbf{D},\mathbf{Y}} \|\mathbf{A} - \mathbf{X}\mathbf{D}\mathbf{Y}^T\|_F$$

- SVD: $\mathbf{X}$, $\mathbf{Y}$ have orthogonal columns.
- SDD: $\mathbf{X}$, $\mathbf{Y}$ have elements in $\{0, 1, -1\}$.
- non-negative factorizations: $\mathbf{X}$, $\mathbf{Y}$ have nonnegative elements.

As in the SDD, the solution is not unique.

Cost of Non-negative Factorization

- The computation is done using an alternating iteration:

  find a better $\mathbf{X}$,
  and then find a better $\mathbf{Y}$,
  at a cost of O(mnp) per iteration.
- Updating can be done at the cost of additional iterations.
- Sparsity can be enforced by adding constraints (Hoyer; Mu, Plemmons, Santago).

Uses of Non-negative Factorization

- LSI

Software and References

Software

For computing the SVD and solving matrix problems in Fortran or C, look for LAPACK software (more than 20 million downloads!). For Java, see `http://math.nist.gov/javanumerics/`. These systems provide

- numerically stable algorithms.
- a uniform interface, making use easy.
- row or column oriented implementation, appropriate for the matrix storage scheme used by the language.
- software built using BLAS (Basic Linear Algebra Subroutines, tuned for a specific machine) and thus are efficient.

For sparse SVD: SVDPack (Michael Berry).

For the URV: Pete Stewart.

For the SDD: look for SDDPACK on my homepage.

For a nonnegative decomposition: see paper by Shahnaz, Berry, Pauca, Plemmons.

References for Fundamentals

For the SVD, eigendecomposition, and QR:

- Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Third edition, Johns Hopkins University Press, Baltimore, MD, 1996.
- G. W. Stewart, *Matrix Algorithms Volume 1: Basic Decompositions*, SIAM Press, Philadelphia, 1998.

An Application of Matrix Methods: QCS

Coauthors:

- Daniel M. Dunlavy, Sandia National Laboratory
- John M. Conroy, Center for Computing Sciences, IDA
- Judith D. Schlesinger, Center for Computing Sciences, IDA

References: http://stiefel.cs.umd.edu:8080/qcs/

Daniel M. Dunlavy, Dianne P. O'Leary, John M. Conroy, and Judith D. Schlesinger, "QCS: A System for Querying, Clustering, and Summarizing Documents," *Information Processing and Management*, to appear. DOI:10.1016/j.ipm.2007.01.003

QCS

We have designed a novel information retrieval system, the Query, Cluster, Summarize (QCS) system. This system is

- portable and
- modular,

permits experimentation with different instantiations of each of the components.

Most importantly, having three components in the QCS design improves retrievals by providing more focused information to the user.

Given a query, QCS

- Retrieves relevant documents.
- Separates the retrieved documents into topic clusters.
- Creates a single summary for each cluster.

In the current implementation,

- Latent Semantic Indexing is used for retrieval,
- Generalized spherical k-means is used for the document clustering, and
- A hidden Markov model and a pivoted QR decomposition is used for summarizing each cluster.

We have used QCS for information retrieval in two information domains:

- newswire documents from the Document Understanding Conferences (DUC)
- documents from Medline (in collaboration with Timothy O'Leary)

Our system demonstrates

- the feasibility of assembling an effective IR system from existing linear algebra software libraries,
- the usefulness of the modularity of the design,
- the value of this particular combination of modules.

A Motivating Example

There are millions of documents on the World Wide Web pertaining to Michael Jordan.

Most of these concern the basketball star, so it is difficult to find information about

- the television personality,
- the jazz musician,
- the mathematician,
- the many others who share that name.

It would be useful to have a system that could overcome this limitation.

The Value of Clustering and Summarizing

One approach is to cluster the documents after retrieval and present a summary of each cluster so that a user can choose clusters of interest.

This is the motivation for our Query, Cluster, Summarize (QCS) system.

Relation to Other Work

- Radev, Fan, Zhang 2001 review previous work on using clustering and summarization to improve IR.
- Of existing IR systems employing this combination, QCS most resembles the NewsInEssence system (CLAIR group, University of Michigan). Both systems can produce multidocument summaries from document sets clustered by topic.

  NewsInEssence: HTML-linked document sets
  QCS: generic document sets.
- The Columbia Newsblaster, like NewsInEssence, is a web-based system that crawls news websites and then clusters and summarizes the news stories, but it does not currently accept queries.
- Recently, Maña-López, de Beunaga, Gómez-Hidalgo 2004 showed increases in user recall of retrieved information when clustering and summarization were included in the output of the IR system.

The QCS System

- QCS is a collection of software modules developed in the C and C++, with preprocessing in Perl.
- QCS has been developed as a client-server application, and the implementation took approximately 6 person-months of full-time effort.

Document Preprocessing

In preprocessing a document set for use with QCS, we

- convert the documents to a standardized format,
- determine the parts of speech (POS) of all words,
- detect and mark the sentence boundaries,
- classify sentences by their content, and
- develop a compact representation for the document information, including a term-document matrix.

Notes:

- Preprocessing and postprocess are the only modules that don't completely rely on linear algebra!
- Recent experiments (DUC-2006) indicate that sentence splitting can be effectively performed without POS tagging.

Indexing of Terms and Documents

The indexing of the terms and documents is performed in QCS using the Berry's General Text Parser (GTP).

- GTP also produces the term-document matrix and its SVD for use in the querying module.
- GTP was chosen for use in QCS since it is a full implementation of Latent Semantic Indexing (LSI) for query-based retrieval, including document parsing, term indexing, SVD computation, and a query tool.
- Minor changes were necessary to provide an interface to the the term-document matrix consistent with that needed by the clustering module.

- The indexing is done "offline"; it is performed once as a preprocessing step for a static document set or during system downtime for a dynamic set.

Queries

- We represent a query using a query vector, $\mathbf{q}$ with m components, just as a document can be represented by a feature vector that forms a column of $\mathbf{A}$.
- A query vector is typically much more sparse than a feature vector and does not necessarily use the same scaling scheme.
- For comparing query vectors and document vectors in LSI, the query vector is projected into the p-dimensional subspace spanned by the columns of $\mathbf{A}_p$, and we denote the projected vector as $\mathbf{q}_p$.
- The relevance of a document to a query is measured by the cosine similarity score s between $\mathbf{q}_p$ and the column of $\mathbf{A}_p$ corresponding to that document:

$$s_j = \frac{\mathbf{q}_p^T (\mathbf{a}_p)_j}{\|\mathbf{q}_p\| \, \|(\mathbf{a}_p)_j\|} ,$$

  where $(\mathbf{a}_p)_j$ is the jth column of $\mathbf{A}_p$.

- In the current implementation of QCS, $N = 100$ documents are returned in order to have a large enough subset of the documents to guarantee good clustering and summarization output.
- Thus some of the retrieved documents may have very low query scores.

This may need to be adjusted based on the document set.

Clustering Documents

Our clustering of the N documents, $\{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$, partitions into k disjoint subsets, $\pi_1, \ldots, \pi_k$, based on cosine similarity of the feature vectors.

The coherence of the cluster π_j can be defined as

$$\sum_{\mathbf{d}_i \in \pi_j} \mathbf{d}_i^T \mathbf{c}_j ,$$

where $\mathbf{d}_i$ is assumed to be normalized (i.e, $\|\mathbf{d}_i\| = 1$) and $\mathbf{c}_j$ is the normalized centroid of cluster π_j containing n_j documents:

- We choose the clusters π_j to maximize the sum of the coherence functions.
- This can be shown to be equivalent to minimizing the radii of the clusters.
- We use the spherical $k-$means algorithm GMEANS (Dhillon, Fan, Guan 2001).

Initializing the Clustering

- Clustering can be a computational bottleneck unless a good initial guess is provided.
- In QCS, we choose an initial partitioning based on the query scores (i.e., the cosine similarity scores). We use 5 initial (seed) clusters and allow the results to be partitioned into as many as $N/10$ final clusters.
- The documents are initially partitioned into 5 equal-sized clusters, based on their query scores.

Advantages of GMEANS:

- efficient computation of feature vector similarities (the main computational bottleneck in many implementations of k-means algorithms).
- the ability to choose a range for the number of clusters into which the feature vectors will be partitioned.

Summarizing Documents and Clusters

The summarization module in QCS is based on a system we used in the DUC-2003 competition.

- choose sentences to include in the summary.
- trim the sentences.

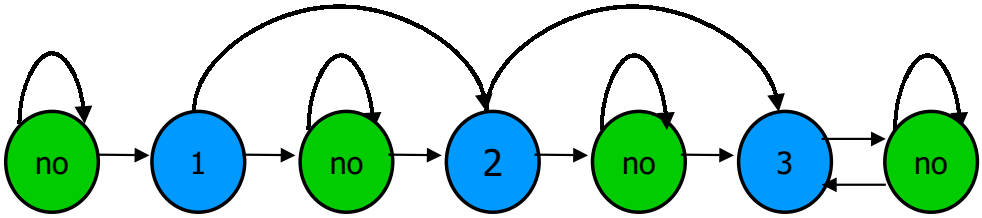
Choice of Summary Sentences

- First, single document extract summaries are produced for each document in the cluster.

- Then, sentences from these summaries are considered for inclusion in the summary of the document cluster.

Single-document Summaries

QCS uses a hidden Markov model (HMM) to score the sentences for inclusion in a summary.



The 13-state HMM extracts six *primary* sentences and an arbitrary number of additional supporting sentences. The HMM uses features based upon

- signature terms, those more likely to occur: $\log(n_{\text{sig}} + 1)$. in the document (document set) than in the corpus at large.
- subject terms that occur in the sentence: $\log(n_{\text{subj}} + 1)$.
- position of the sentence in the document, built into the state structure.

More recently, we have used a linear-algebra-based approximate Oracle score in place of the HMM. The oracle estimates the probability that each term in the sentence is a term that a human would choose to be in a summary.

In any case, the highest probability sentences are chosen for the summary.

Multidocument Summaries

Multidocument summaries are created for each cluster by choosing a subset of the sentences identified by the HMM.

- If we want a summary containing w words, we consider the highest probability sentences from documents in that cluster, cutting off when the number of words exceeds $2w$.
- We form a term-sentence matrix, $\mathbf{B}$. The columns of $\mathbf{B}$ are scaled so that the Euclidean norm equals the score assigned to the sentence by the HMM or approximate Oracle.
- In order to remove redundant sentences, a pivoted QR algorithm is applied to the scaled term-sentence matrix.

Sentence Trimming

Longer sentences tend to have higher scores.

- Because of this, summaries would have a very small number of sentences.
- We decided to trim sentences to try to remove unimportant information from them.

As an inexpensive alternative to full parsing and comprehension, we identified trimming patterns using shallow parsing techniques, keying off of lexical cues.

- lead adverbs and conjunctions;
- gerund phrases;
- restricted relative-clause appositives;
- intra-sentential attribution.

The QCS Client-Server Architecture

There are three main frames in the user interface:

- The query form contains an input field for entering a query and a field for selecting the document set on which to perform the query.

- The navigation bar contains links to the documents and is organized to reflect the output from the querying, clustering and summarization modules.
- The results frame displays information requested through the navigation bar.

Example of the QCS System

This example uses the query hurricane earthquake in finding documents in the DUC 2002 document collection.

The DUC 2002 collection consists of 567 documents and the QCS preprocessing modules identified 7767 unique terms across the collection.

A view of the screen ...

Result of Querying

Score	*Subject Line*
.90	Hurricane Latest in String of Disasters to Hit Historic City
.85	Hurricane Forecasters Carry on Amid Chaos
.85	Forecasting Aided by Supercomputers, but Still an Uncertain Science
.84	Killer Storm Hits South Carolina Coast
.83	Scientists: Warming Trends Could Mean Fiercer Hurricanes
.82	City Sends Money to Charleston in Repayment of 211-year-old Debt
.82	150,000 Take Off as Hugo Takes Aim at Ga., Carolina
.82	Loss of Life Low because People Were Prepared
.81	Hurricane Gilbert Heading for Jamaica with 100 MPH Winds
.80	Gilbert: Third Force 5 Hurricane This Century

Result of Clustering

Cluster	Documents	Mean Query Score
1	19	.72
2	15	.70
3	11	.51
4	15	.41
5	17	.34
6	6	.20
7	8	.17
8	3	.17
9	3	.13
10	3	.08

Score	Subject Line
Cluster 1	
.83	Hurricane Gilbert Heading for Jamaica With 100 MPH Winds
.80	Gilbert: Third Force 5 Hurricane This Century
.80	Hurricane Hits Jamaica With 115 mph Winds; Communications Disrupted
Cluster 2	
.83	Forecasting Aided By Supercomputers, But Still An Uncertain Science
.83	Hurricane Latest in String of Disasters to Hit Historic City
.79	Hurricane Forecasters Carry On Amid Chaos
Cluster 3	
.67	Hurricane batters southern US but lets insurers off lightly
.67	US insurers face heaviest hurricane damage claims
.66	UK Company News: GA says hurricane claims could reach 'up to Dollars 40m'

Result of Summarizing

Cluster 1

Gilbert, an "extremely dangerous hurricane" and one of the strongest storms in history, roared toward Mexico's Yucatan Peninsula Tuesday with 175 mph winds after battering the Dominican Republic, Jamaica and the tiny Cayman Islands.

At midnight EDT Gilbert was centered near latitude 21.5 north, longitude 90.2 west and approaching the north coast of Yucatan, about 60 miles east-northeast of the provincial capital, Merida, the National Hurricane Center in Coral Gables, Fla., said.

John Hope, the network's hurricane specialist and a former forecaster with the National Hurricane Center in Miami, Fla., said the drought in the Southeast might be lessened or ended in the next few months by a heavier than normal hurricane season.

Cluster 2

Hurricane Hugo advanced faster and with renewed fury today on Georgia and South Carolina as 150,000 coastal residents grabbed what they could carry and fled inland on jammed highways.

Supercomputers, satellites and the expertise of several hurricane forecasters predicted the destructive path Hurricane Hugo would follow, giving people plenty of time to flee the South Carolina coast.

The storm, which caused billions in damage, claimed 17 lives in South Carolina, and only two were in the Charleston area, which bore the brunt of Hugo's 135 mph winds.

While Hurricane Hugo's 135 mph wind roared outside, Mayor Joseph P. Riley Jr. watched the fury it vented on his beloved, 300-year-old city.

Cluster 3
Hurricane Hugo will go down in the record books as the costliest storm insurers have faced so far, but it won't cause property-casualty premium rates to rise immediately, analysts and company officials say.

Most San Francisco-area homeowners may have to pay for damage from Tuesday's earthquake out of their own pockets, while insurance companies may reap long-term benefits from higher rates, industry spokesmen and analysts said Wednesday.

Although the damage from the hurricane's landfall in Florida on Monday was much greater than initially estimated, insurers' losses there are likely to total less than Dollars 1bn, well below earlier expectations, a senior member of Lloyd's insurance market said yesterday.

Cluster 4
A major earthquake rocked northern California Tuesday evening, collapsing part of the San Francisco Bay Bridge and shaking Candlestick Park and buildings up to 95 miles away.

Tuesday's earthquake, the strongest on the San Andreas fault since the San Francisco quake on April 18, 1906, came in a place that had been identified by scientists just last year as the most likely spot for a major jolt in Northern California within the next 30 years.

A violent earthquake rocked Northern California during Tuesday evening's rush hour, caving in a section of the San Francisco Bay Bridge, terrifying World Series fans in Candlestick Park and shaking buildings as far as 200 miles away.

Cluster 5
Iran said today that it would welcome relief offered by its bitter enemy, the United States, to help victims of the earthquake that has killed as many as 35,000 people, the State Department said in Washington.

State Department officials said the government gave $300,000 worth of supplies to the American Red Cross for shipment to Iran – including 1,000 hard hats, 1,000 pairs of leather gloves, 10,000 face masks, 2,940 wool blankets and about 500 tents.

Orange County's Iranian community launched an ambitious effort Sunday to collect half a million dollars in money, medicine, tents, blankets and sleeping bags for hundreds of thousands of injured and homeless people in earthquake-ravaged Iran.

Conclusions

- There are many matrix decompositions that have proved useful in

information retrieval, including

  - Eigendecomposition, useful in PageRank.
  - Pivoted QR, useful for redundancy reduction.
  - SVD, useful in LSI.
  - URV, useful in LSI.
  - SDD, useful in LSI and compression.
  - non-negative factorizations, useful in LSI.

- Improving algorithms for computing and updating these decompositions would greatly improve the efficiency of information retrieval.
- It is amazing how much of an information retrieval system can be built just from matrix algebra!