Text Document Cluster Analysis Through Visualization of 3D Projections

Masaki Aono and Mei Kobayashi

Abstract Clustering has been used as a tool for understanding the content of large text document sets. As the volume of stored data has increased, so has the need for tools to understand output from clustering algorithms. We developed a new visual interface to meet this demand. Our interface helps non-technical users understand documents and clusters in massive databases (e.g., document content, cluster sizes, distances between clusters, similarities of documents within clusters, extent of cluster overlaps) and evaluate the quality of output from different clustering algorithms. When a user inputs a keyword query describing his/her interests, our system retrieves and displays documents and clusters in three dimensions. More specifically, given a set of documents modeled as vectors in an orthogonal coordinate system and a query, our system finds three orthogonal coordinate axes that are most relevant to generate a display (or users may choose any three orthogonal axes). We conducted implementation studies to demonstrate the value of our system with an artificial data set and a de facto benchmark news article dataset from the United States NIST Text REtrieval Competitions (TREC).

1 Introduction

Clustering has been used to understand the contents of databases and for information retrieval [12, 20, 27, 35]. Many established algorithms are *hard clustering* methods that model the data space as a set of disjoint clusters with background

M. Aono
Department of Computer Science and Engineering, Toyohashi University of Technology , 1-1 Hibarigaoka, Tempaku-cho, Toyohashi-shi, Aiichi 441-8580, Japan
e-mail: aono@tut.jp

M. Kobayashi (✉)
IBM Research-Tokyo, 5-6-52 Toyosu Koto-ku, Tokyo 131-8511, Japan
e-mail: MeiKobayashi@gmail.com

K. Yada (ed.), *Data Mining for Service*, Studies in Big Data 3,
DOI: 10.1007/978-3-642-45252-9_15,

noise. Partition-based hard clustering techniques require all documents to belong to exactly one cluster. Examples include: K-means [29], K-medoid, and agglomerative (hierarchical) methods.

As databases have increased in size and complexity, *soft* clustering methods that permit cluster overlaps have been developed [4, 7, 11, 24, 37, 45]. These methods lead to more meaningful clusters. For example, consider medical document sets. An article on *alcohol* and *liver disease* will be placed in only one cluster (not both) by hard clustering methods, while soft clustering methods will place the article in both the *alcohol* and *liver disease* clusters. Soft clustering methods are also useful for scientific analysis of micro-array expression data since they permit genes to belong to several clusters. This reflects a more accurate view of biological processes since most genes have multiple functions and participate in multiple reaction pathways [5, 15, 40, 46].

Understanding output from clustering algorithms may be difficult for massive data bases replete with overlapping, heterogeneous-sized clusters. Methods for cluster labeling are reviewed in [34]. The high dimensionality of the data sets poses an additional challenge. Several types of structures for interfaces have been proposed, e.g., charts, spirals, tree structures, and annotations, e.g., [39, 41–44]. Visualization using graphics is a simple approach that allows non-technical users to quickly understand data with limited training. We review visualization interfaces related to ours that influenced our work.

1.1 Related Work

The idea of analyzing large, multivariate data sets by finding appropriate low dimensional bases or slices and displaying them was first proposed by [28]. The first successful implementations were conducted by [14], whose term "*projection pursuit*" has since been widely adopted. Their aim was to develop a system to automatically find "*interesting*" linear projections of multivariate data onto a line or plane. Nason [31] extended the projection pursuit concept and implemented a system to find and visualize information rich 3D subspaces of multidimensional data.

Reference [17] proposed *HD-Eye*, a system that clusters high dimensional data by taking a series of lower dimensional projections. The main idea of their algorithm is: if each projection identifies *some* partitions, then the combined information from a sufficient number of projections will enable identification of all clusters. A projection does not have to find all partitions, just some. Directions that are likely to yield fruitful partition data are computed by the system using kernel density estimation techniques. Success of the method requires no false identification of partitions (that do not exist). The weaknesses of this method include: the possibility of these false positives; difficulty in determining when a sufficient number of partitions have been found; and lack of stopping criteria (*HD-Eye* is based on a genetic algorithm, so users must have some knowledge of the properties of the data in order to decide when to stop). Another lower-dimensional visualization by [18] outputs 2D results

of *Fastmap*. Visualization of lower dimensional slices of large multivariate data sets using classification (rather than clustering) is reported in [10].

An alternate approach for viewing contents of data bases in lower dimensions is the use of *several* (i.e., more than three) coordinate axes that are not mutually orthogonal to display information about documents, for example, *Parallel Coordinates* [19]. Another example is a star-like circular plot [2] for visualizing multidimensional data with multicolored circle segments (i.e., sets of radial lines from the center to the perimeter of a circle) to represent database contents. *Star Coordinates* [22] is a visualization interface based on circular plots for cluster analysis that displays relationships between documents and attributes in 2-dimensional space. Each attribute is represented as a vector emanating from the origin and each document as a square dot. Users can understand topics and clusters through scaling and rotation of the attributes. As attribute axes are moved, the document points are dynamically repositioned. *Star Coordinates* is a tool for projecting documents from a very high dimensional attribute space into a lower dimensional space to facilitate topic, trend, and cluster analysis. It does not use orthogonal projections and does not have a system to recommend the best viewing angles. However, it allows projections that involve more than three attribute dimensions. Since *Star Coordinates* do not match our natural, innate 3-dimensional human visual system, it is difficult for users to judge whether documents in a cluster are closely or loosely related.

The method in our prototype to display clusters is similar to that of [36], which retrieves matching terms from documents with a high relevancy ranking for queries, and uses this information together with the ranked document list to generate a 3-dimensional blobby image with documents and terms attached as 3-dimensional text-based annotation. Some problems with their system are: it relies on the *Smart* search engine [38], whose ability to process massive sets of data is dubious; distances between term vectors in 3-dimensional space do not necessarily correspond to the relevance between matching terms; blobs are not a good way to display information using computer graphics because annotated text can be obscured by the blob; and protrusions from a blobby object tend to be relatively thin compared to the rest of the blob even though they have a larger term weight. Our prototype overcomes these problems.

1.2 Our Work and Contributions

We developed a visualization system with the following new features. Given any multidimensional, orthogonal coordinate system, users input keywords describing a topic of interest (as they would when using search engines, such as $Google^{TM}$, $Yahoo!^{TM}$ or $Bing^{TM}$) to generate a list of appropriate axes for displaying documents—or users may select any three coordinates. Our system identifies documents and clusters with a significant presence in the subspace defined by the three axes, then displays a projection of them in the three-dimensional space. Keyword labels are automatically generated for the clusters. When invoked, optional pop-up windows will display a list of titles and relevancies of identified documents as well as the entire text of

individual articles. Standard equipment on PCs are used for rotations and manipulations of the view to facilitate navigation of data and observation from different angles and distances by non-technical users.

Our system enables users to see only data that is relevant to their interests and to see properties of the data, such as: sizes of clusters, distances between clusters, extent of cluster overlap(s), similarities of documents within a cluster (i.e., how tightly or loosely the data objects are bound to one another). Our system can also be used to see results from other clustering algorithms (not just ours) if orthogonal coordinate vectors are provided. The GUI is a useful tool for evaluating the quality of output from different algorithms. Screen shots from our system were published in previous works on identification of possibly overlapping, small (minor) clusters in databases [23–25]. However, the works did not mention the visualization algorithms and system design.

The remainder of this chapter is organized as follows. The next section reviews algorithms from information retrieval used in our system: *vector space modeling* (VSM), *latent semantic indexing* (LSI), *covariance matrix analysis* (COV), and extensions of these algorithms—*LSI with rescaling* and *COV with rescaling*—that are better at finding smaller, *minor* clusters. The third section presents algorithms that enable real-time 3D-visualization of search results. The algorithms: (1) accept keyword inputs from users and generating a ranked list of coordinate axes (from LSI or COV) for displaying relevant documents and clusters of in a massive data set, (2) find and display documents with significant components in a subspace spanned by three coordinate axes, and (3) compute (possibly overlapping) clusters in the subspace, and display them in contrasting colors. The fourth section discusses implementation studies using our visualization system and artificial and real-world data sets. We show images of overlapping heterogeneous-sized clusters in 3-dimensional subspaces. The fifth and final section summarizes our findings and points to directions for future research.

2 Coordinate Transformations for 3D Projection and Display

2.1 Vector Space Modeling(VSM)

Vector space modeling (VSM) of documents has become commonplace in database analysis since its introduction over three decades ago [6, 38]. Keywords are used as attributes (represented as coordinates) for constructing vector models of text documents. Implementation studies with our visualization system are based on VSM of text documents using *Term Frequency-Inverse Document Frequency* (TF-IDF) term weighting which accounts for the relative importance of a term in a document (with respect to other terms in the document) as well as the importance of the term in other documents. The query input describing user interests is modeled as a vector, and the relevancy ranking of coordinate and document vectors with respect to the query depends on the *cosine distance*, (the cosine of the angle defined by a query and document vector) to the query vector [3, 6].

2.2 Latent Semantic Indexing and Covariance Matrix Analysis

The number of documents in modern databases is so massive that straightforward use of distance computations for real-time search and retrieval of documents is impossible. One approach for solving this problem (known in the data mining community as, "*the curse of dimensionality"*) is to project it into a subspace of sufficiently small dimension to enable fast response, but large enough to retain distinguishing characteristics of individual documents. Two well-established algorithms for carrying out dimensional reduction are *latent semantic indexing* (LSI) [9] and *covariance matrix analysis* (COV) [21, 30, 33]. (Note: COV is known as *principal component analysis* (PCA) in the image processing community, where the matrix is considerably smaller so computing eigenvectors is simpler).

Both LSI and COV generate orthogonal linear transformations that map the coordinate system for VSM to a new orthogonal coordinate system so that the first coordinate (*singular vector* for LSI, or *principal component* for COV) contains the most information about the data set as a whole, the second coordinate has the second most information, and the ith coordinate has the ith most information.

LSI is based on VSM of documents in a database. The relationship between all M documents in a data set and N keywords that appear in the documents is represented as an M-by-N, rectangular matrix $\mathbf{A} = (a_{ij})$. The LSI algorithm computes the *singular value decomposition* (SVD) of **A,** that is, $\mathbf{A} = \mathbf{U\Sigma V^T}$, where **U** and **V** are M-by-M and N-by-N unitary matrices, and $\mathbf{\Sigma}$ is a diagonal M-by-N, matrix with diagonal elements σ_i; known as *singular values* [16]. The singular vectors are used for coordinate transformation and visual display of documents and clusters. The partial SVD is normally used in practice; Only the largest several hundred singular values and singular vectors.are computed.

COV is based on computing the eigenvectors (a.k.a. *principal components*) of the covariance matrix associated with a set of document vectors. Only eigenvectors associated with the largest several hundred eigenvalues are used for coordinate transformation and visual display.

Our system can use any orthogonal coordinate system for projection and display and is not necessarily restricted to those generated by LSI and COV. For example, some other approaches for dimensional reduction for analysis of massive document sets are: random projections [1, 13]; centroid and least squares analysis [32]; and a Krylov subspace method by [8].

References [24, 26] developed efficient, new methods specifically for identifying small, *minor* clusters. They noted that while LSI and COV are effective for finding large, *major* clusters in a dataset, the same major clusters are often repeatedly identified from the largest several hundred singular vectors and eigenvectors. However, minor clusters are identified (if at all) only after a large portion of singular vectors are computed. Since finding each singular value and singular vector requires more computation than finding their predecessors, LSI and COV were found to be inefficient (at best) or ineffective (at worst) for finding minor clusters. The methods by Kobayashi, Aono et al. are extensions of LSI and COV. The main idea is to compute a

singular vector or eigenvector and find clusters near the vector. Before computing the next vector, information about the newly identified clusters is suppressed to prevent its repeated re-discovery. The modified algorithm *LSI with rescaling* is not scalable, i.e., cannot be applied to large data sets, however, *COV with rescaling* is scalable.

3 Dynamic Clustering for 3D Display

Many institutions have massive and dynamic data sets that are impractical or impossible to analyze and cluster in their entirety. We introduce algorithms to search, cluster and label only documents that are of interest to a user.

3.1 Document Retrieval

A summary of the procedure to retrieve documents and identify clusters in our visualization system is given in Fig. 1. In the first step, users input a query (keywords) describing topics of interest. Keyword weighting is optional. The query is converted to a vector, and the system computes three recommended axes for displaying results based on their proximity to the query vector using the cosine distance. This feature is needed because of the huge number of possible choices for axes for the display (for example, in 200-dimensional space, there are (200!/197!3!) possible choices for selecting three axes). Users can customize their choice of axes by asking for a list of rankings of basis vectors, then selecting three axes for the display.

Our system does not use pre-computed clusters. They are generated on-the-fly as documents are retrieved based on the user query. Vectors of retrieved documents have a significant component along the direction of the query vector (LHS, Fig. 2). Two parameters ($theta\ \theta$ and $alpha\ \alpha$) control the filter for noise reduction. Only documents that lie within an angle $theta\ \theta$ of the query vector are retrieved. In typical data sets, many documents that are remotely related to the query lie very close to the origin, within the *theta cone* around the query vector. Display of these documents would lead to meaningless clutter around the origin, so documents that lie within an apex of height $alpha\ \alpha$) of the cone are deleted. Values for $theta\ \theta$ and $alpha\ \alpha$) can be adjusted using a slider bar (RHS, Fig. 2).

3.2 Cluster Identification and Keyword Labeling

We use a two-part soft clustering algorithm: *cluster keyword generation* and *cluster merge and labeling*. We input: several hundred basis vectors from LSI or COV, the number of cluster labels for *extrinsic* and *intrinsic* keywords (defined later), a threshold δ (to separate clusters), and keyword data extracted from documents.

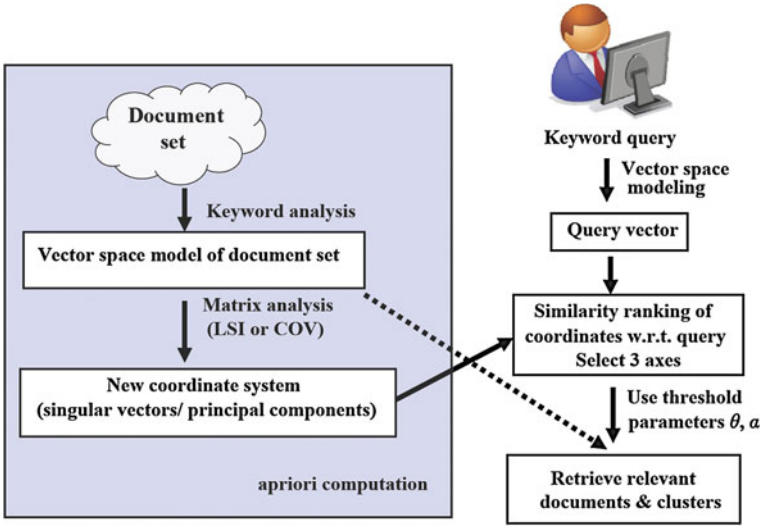

Fig. 1 Overview of process for generating three coordinate axes for displaying documents and clusters in a database in response to a keyword input query. The default setting generates a 3D-subspace display using coordinates with the top three similarity rankings with respect to a query. Only documents and clusters related to the query will appear in the 3D-display

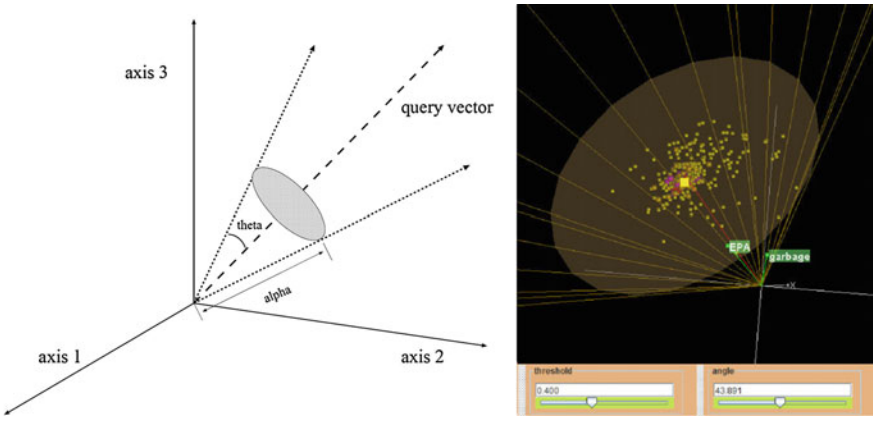

Fig. 2 Threshold parameters to determine documents that will appear in the 3D-display. Documents that lie within an angle theta of the query and have a length greater than alpha qualify. A screen image of cluster on "EPA, garbage" in LA Times news article dataset from TREC appears on the *left*

3.2.1 Cluster Keyword Generation

This algorithm computes the similarity between all basis vectors and all document vectors and produces *extrinsic and intrinsic* keywords when the similarity is greater than a pre-set threshold δ. *Extrinsic* keywords are the top $p(e)$ contributors to the

similarity between a basis and document vector. *Intrinsic* keywords are the top $p(i)$ contributors to the TF-IDF weights in the document vector

Cluster keyword generation:

1. Read in basic vectors (singular vectors or principal components), keywords, documents vectors, stop-word list, and pre-set threshold δ
2. Let i (loop variable) be $i0$ (the starting basis vector Id, which is typically 1).
3. Repeat the following while $i < k$.
   - 3.1. Compute similarity between i-th basis vector and all document vectors.
     - 3.1.1. Keep $p(i)$intrinsic keywords for each document vector.
     - 3.1.2. Keep $p(e)$extrinsic keywords for each document vector.
   - 3.2. Sort in descending order based on similarity.
   - 3.3. For documents with similarity values greater than $\delta > 0$, do the following,
     - 3.3.1. Output cluster candidates & intrinsic & extrinsic keywords from 3.1
   - 3.4. For documents with similarity values less than -$\delta < 0$ do the following,
     - 3.4.1. Output cluster candidates & intrinsic & extrinsic keywords from 3.1

Note:

- In steps 3.3.1 and 3.4.1, all stop-words are eliminated from consideration.
- Keywords that are intrinsic and extrinsic are output as an extrinsic keyword.

3.2.2 Merging and Labeling Clusters

This algorithm uses several data structures. The first three are created when the data file produced in the *cluster keyword generation* program is opened and parsed. *doclist* is generated when parsing the output of *cluster keyword generation* by assigning *docId* to each document Id, *keywordList* to the sum of intrinsic and extrinsic keywords, and *similarity* to the dot product between a basis and document vector. *keyDoclist* is initialized by putting all the intrinsic and extrinsic keywords into a hash table *keyTable*, so that *docId* is appended to *docIdlist* of *keyDoclist*.

The final process is carried out when one of the following conditions is met during parsing of the output: (i) start parsing the output for the next basis vector, (ii) start parsing in the negative (minus) direction for the output of the current basis vector output, or (iii) finish parsing all basis vectors. Keyword merging takes place during insertion sorting (step 5.6.2.1.3). Cluster type (major, minor, noise), is determined by the number of documents in step 5.7. If M is the number of documents, b and c are constants with $b < c$, and n is the number of documents in a cluster, then a major

cluster is defined as ($n > cM/100$); minor as ($cM/100 > n > bM/100$); and noise as ($n < bM/100$). In our studies with *LA Times* news articles, $b = 0.1$ and $c = 1.0$.

Data Structures for Merging and Labeling Clusters:

- *doclist:* // document-keyword list
  docId: integer; // document Id
  similarity: float; // cosine similarity measure
  numKeyword: integer; // number of keywords
  keywordList: array of String // keyword list
  marked: Boolean; // indicates whether document already placed in cluster
- *keyDoclist*: // keyword-docId list
  keyword: String: // keyword
  valur: float; // tracks importance of keyword for cluster label candidacy
  sortIndex: integer; // keeps track of index before sorting
  docIdlist: array of integer; // document Id list
  marked: Boolean; // indicates whether keyword has been processed for labeling
- *keyTable*: // hash table storing *keyDoclist* with keyword as hash key
- *resultId*: // *docId* list for a cluster (for output)
- *resultLabel*: // cluster label (for output)

Merging and Labeling Clusters:

1. With the current *doclist*, scan *keyTable* and get a *keyDoclist* for each keyword.
   1.1 Get a *docIdlist* of *keyDoclist* and see if there is a *docId* in the current *doclist*
   1.2 If a *docId* from *docIdlist* is in the current doclist, add a *value* using the formula: *keyDoclistvalue*+= *f(doclist similarity)*; f is a weighting function. Since the absolute value of the similarity is between 0 and 1, a typical function $f(x) = \alpha x$, where α is positive constant (10.0 by default).

1.3 Repeat this process from 1.1 until all keywords in *keyTable* are scanned.
2. Sort keywords in *keyTable* with indices in descending order based on *value* of *keyDoclist*.
3. Set *sortIndex* in *keyDoclist* by the indices in step 2.
4. Initialize *resultId* and *resultLabel*.
5. Scan sortex keywords from largest *value* and repeat the following:
   5.1. If *value* is less than β (e.g, 2.0) quit this loop
   5.2. If current *keyDoclist* is *marked*, skip keyword and scan next keyword
   5.3. Get a *doclist* from current keyword
   5.4. If number of keywords in $resultLabel < q$ (e.g, 4) add keyword to *resultLabel*
   5.5. *Mark* the current *keyDoclist*
   5.6. With the *doclist* obtained in step 5.3, repeat the following:
      5.6.1. If *docId* in *doclist* in included in *resultId*, then skip document;
      Otherwise, add *dodId* toresultId
      5.6.2. Scan *doclist*.
      5.6.2.1. If there is a *docId* include in the current *keyDoclist*, then:
      5.6.2.1.1. *Mark* the *doclist*
      5.6.2.1.2. Get a *keywordList* from the *doclist*
      5.6.2.1.3. Scan *keywordList*. If there is an unmarked keyword in *keyDoclist*,
      Do insertion sorting of the keyword to *resultLabel*;
      Otherwise, skip to the next keyword until the *keywordList* becomes empty.
   5.7. Output *resultLabel* (as label for cluster) and *resultId* (as a list of document Ids).
   5.8. If *keyDoclist* is empty, exit the loop;
   Otherwise go back to step 5.1 with the next keyword.

3.2.3 Display of Documents and Clusters in 3D Space

Figure 3 is an overview of the GUI of the system to display relevant documents and clusters as objects in 3D space on a computer screen. Figure 4 shows how the documents (represented as squares) are projected onto a 2D screen.

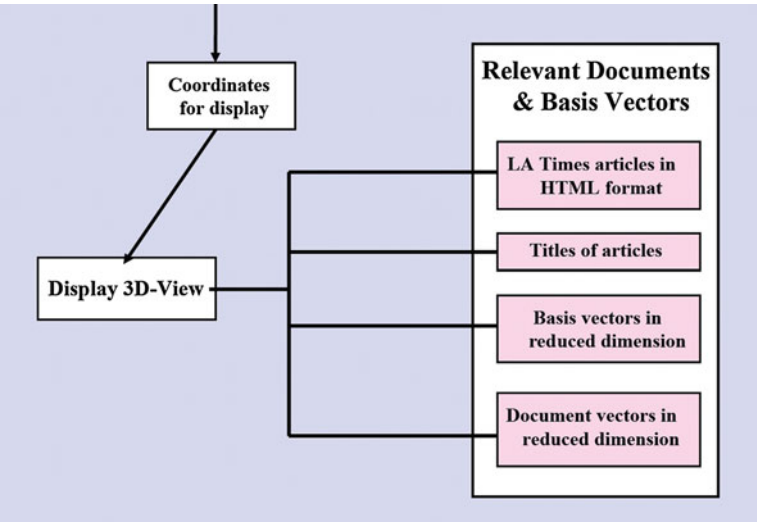

Fig. 3 Overview of graphical user interface (GUI) for our prototype visualization system

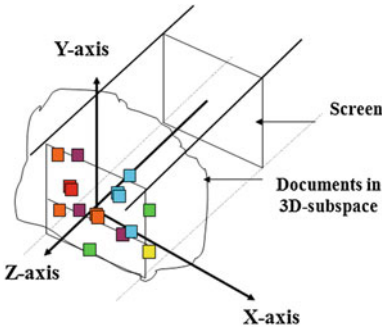

Fig. 4 Projecting documents in 3D-subspace onto a 2D screen to generate a screen view

4 Implementation Studies

We conducted implementation studies of our system with four algorithms (LSI, COV, LSI with re-scaling, COV with re-scaling) and two data sets: artificially generated data to enable verification of results, and a de facto benchmark dataset of 147,742 *LA Times* news articles from *Text Retrieval Competitions* (TREC) (http://trec.nist.gov) to demonstrate viability of analysis of real-world data.

Table 1 Test input data (140 documents, 40 keywords)

25 documents	**Clinton cluster**	**Major cluster**
10 documents	(Clinton + Al Gore)	(Subcluster)
5 documents	(Clinton + Hillary)	(Subcluster)
10 documents	(Clinton + Al Gore+ Hillary)	(Subcluster)
25 documents	**Java cluster**	**Major cluster**
10 documents	(Java + JSP)	(Subcluster)
5 documents	(Java + Applet)	(Subcluster)
10 documents	(Java + JSP+ Applet)	(Subcluster)
5 documents	**Bluetooth cluster**	**Minor cluster**
5 documents	**Soccer cluster**	**Minor cluster**
5 documents	**Matrix cluster**	**Minor cluster**
5 documents	**DNA cluster**	**Minor cluster**
70 documents		**Noise**

Table 2 Clusters identified using subspace singular vectors b_i

Vector	**LSI**	**COV**	**LSI + rescaling**	**COV + rescaling**
b_1	Clinton	Clinton, Java	Java	Clinton, Java
b_2	Java	Clinton, Java	Clinton	Noise, all minor clusters, Clinton, Java
b_3	Noise	Noise, all minor	Noise	Matrix, DNA
b_4	Clinton	All minor, noise	Matrix, DNA	All minor clusters
b_5	Java	All minor, noise	Bluetooth, Soccer	Bluetooth, Soccer
b_6	Noise	All minor, noise	All minor clusters	Noise, all minor clusters

4.1 Artificially Generated Data Set

We constructed an artificial data set with 140 documents, 40 keywords, and non-intersecting clusters as follows: 2 major clusters (each with 3 subclusters), 4 minor clusters, and noise (Table 1). It is modeled on natural databases which typically consist of many, heterogeneous sized clusters and 50 % noise. Results from our studies are summarized in Table 2 and were confirmed using our visualization system. LSI only finds major clusters and noise (Fig. 5a, b), while all major and minor clusters and noise are retrieved by the COV algorithm (Fig. 6a, b).

4.2 LA Times Data Set from TREC

We conducted implementation studies with a real-world dataset consisting of 147,742 *LA Times* news articles using coordinate axes generated by COV. Figure 7 is a screen snapshot from a session in which the user is interested in the topics *baseball, basketball* and *game*, weighted 3,2, and 1, respectively. The three

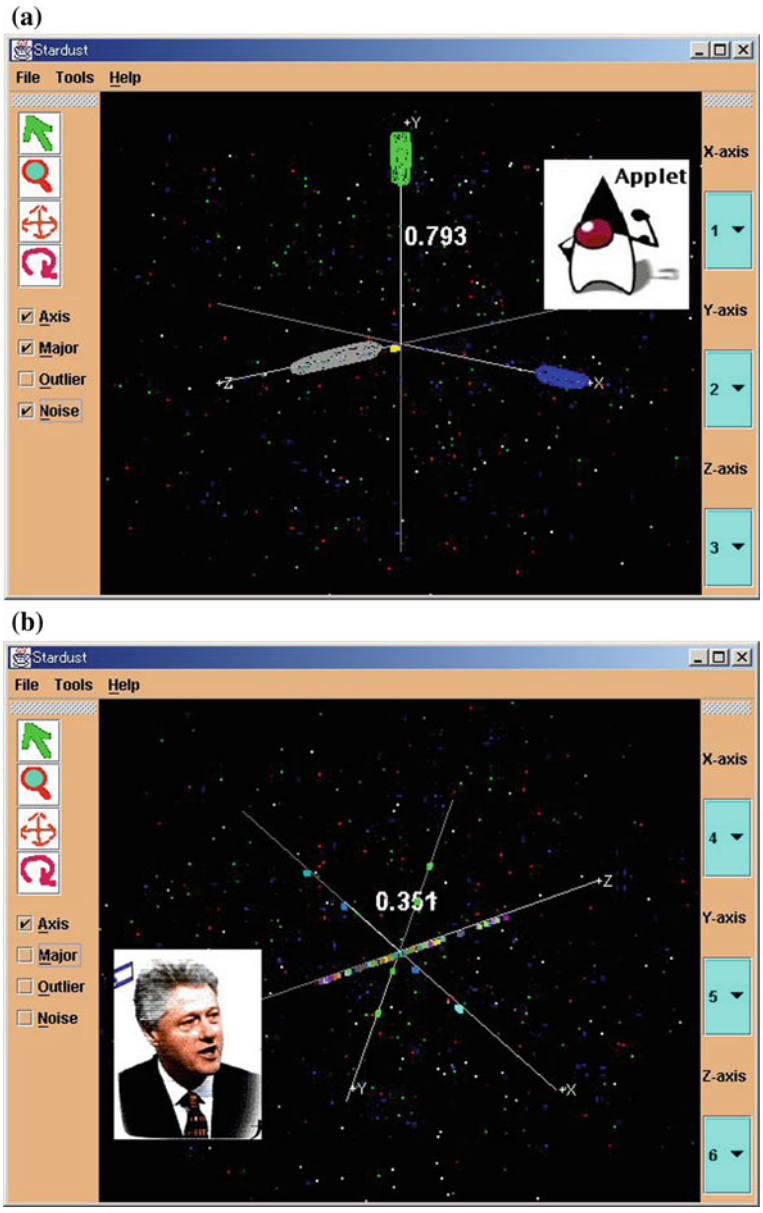

Fig. 5 **a** Only major clusters are seen using LSI. In the subspace spanned by the top three singular vectors, the convex hull of the Java cluster (*green*) lies along the y-axis, and that of the Clinton cluster (*blue*) lies along the x-axis. Noise can be seen along the z-axis. **b** Only major clusters are seen using LSI. In the subspace spanned by the fourth, fifth and sixth singular vectors, only a major cluster on Clinton is seen (no minor clusters)

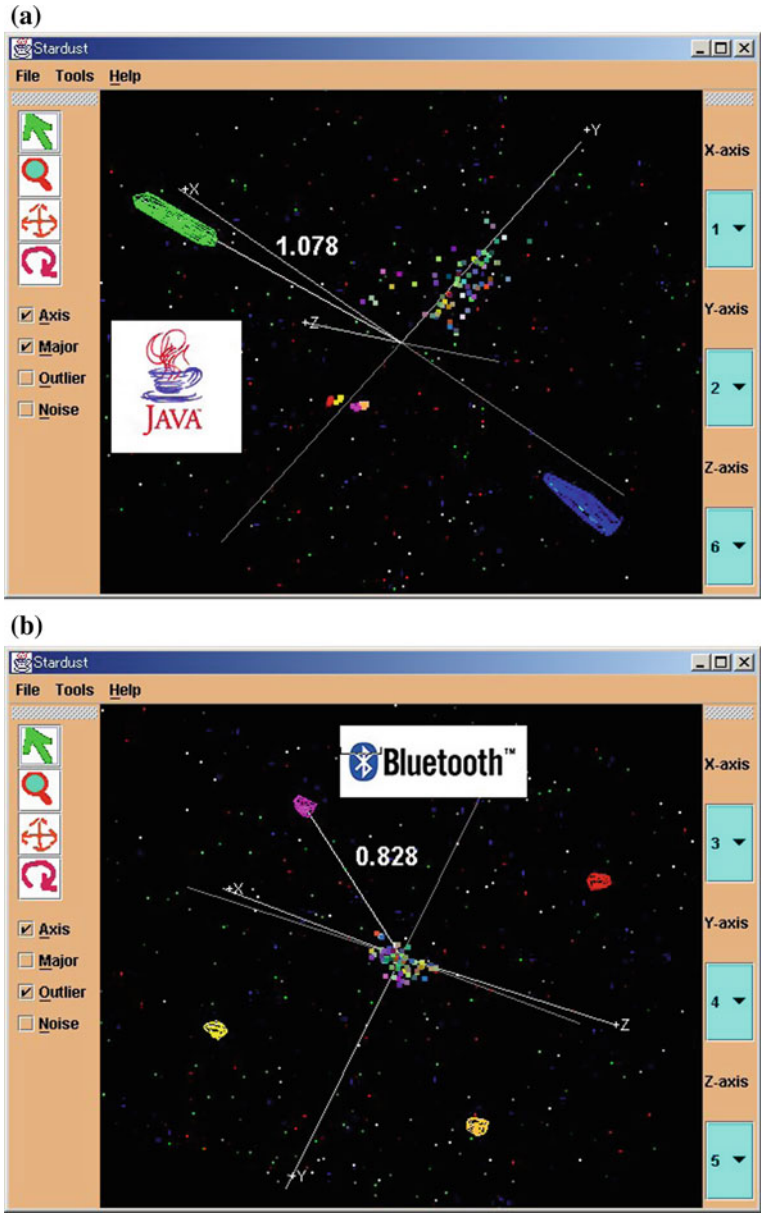

Fig. 6 **a** Major and minor clusters are detected using COV-rescale, a soft clustering algorithm based on covariance matrix analysis with re-scaling by [24]. The Clinton and Java major clusters are seen along the x-axis in the subspace spanned by the first, second and sixth singular vectors. Noise is dominant along the y-axis, and fragments of minor clusters are seen along the y- and z-axes. **b** Four minor clusters identified using COV-rescale using the third, fourth and fifth eigenvectors: Bluetooth (*magenta*), soccer (*red*), matrix (*orange*), and DNA (*yellow*)

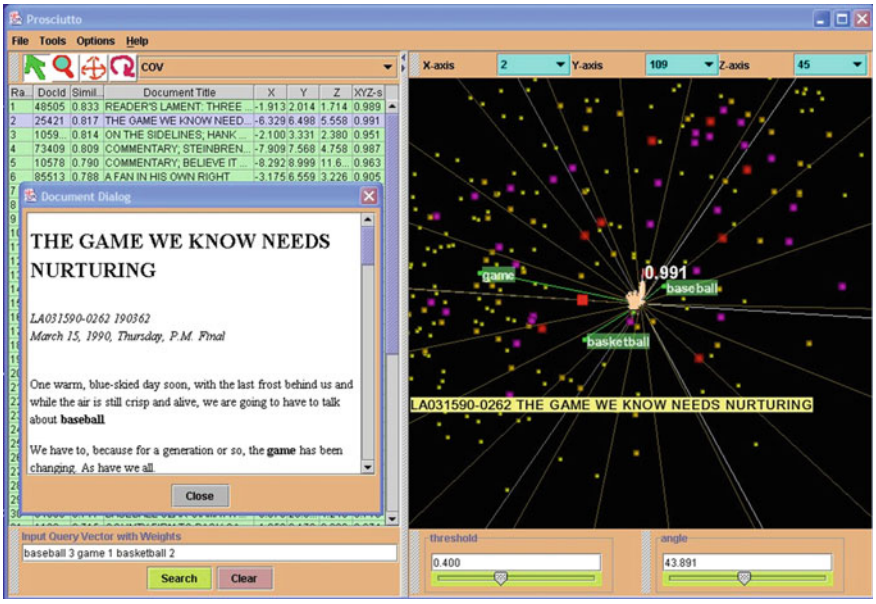

Fig. 7 The query, "*baseball 3 game 1 basketball 2*" (terms and weights) appears in the *bottom*, *left dialog box*. Search results (titles of articles) appear on the *left*, *rear panel*. The second ranked article turns purple when selected from the list, and the full text appears in a pop-up window (*left*, *front panel*) with query terms in boldface (i.e., baseball, game). Basis vectors 2(x-axis), 109(y-axis), and 45 (z-axis) are used for the 3D-display (top right). Vectors for query terms appear in *green*, and articles on baseball, game and basketball appear in *purple*, *red* and *yellow*, respectively. The sliders on the *bottom*, *right* set parameters theta and alpha (Fig. 2)

highest ranking principal components with respect to the query are 2, 109 and 45. They appear as the *x*-, *y*-, and *z*-axes.

In a second example, the user inputs the terms *baseball* and *basketball* with equal weights (Fig. 8). Our system recommends basis vectors 2, 159, and 165 as three axes for display. Basis vector 2 is associated with a cluster of documents on {*team, coach, league, inning*} with 15,631 documents, basis vector 159 with a cluster on {*Aztecs, Arizona, sport, baseball, basketball*} with 330 documents, and basis vector 165 with a cluster on {*baseball, Edison, football, France, Aztecs*} with 621 documents. These cluster keywords suggest that our system's automatic recommendation of basis vectors 2, 159, and 165 for projection and visualization are reasonable for the user query {*baseball 1, basketball 1*}.

Another type of interface for displaying information about clusters is shown in Figs. 9, 10. Figure 9 shows overlapping clusters on {*school, council, tax, valley, California*} and {*school, art, music, film, child*} with 3,433 and 630 documents, respectively. 414 documents belong to both, with overlap ratios 12 and 75.7 %, respectively. Figure 10 shows six outlier clusters lying along basis vectors 58, 63 and 104. Clusters may lie along or in-between positive and negative

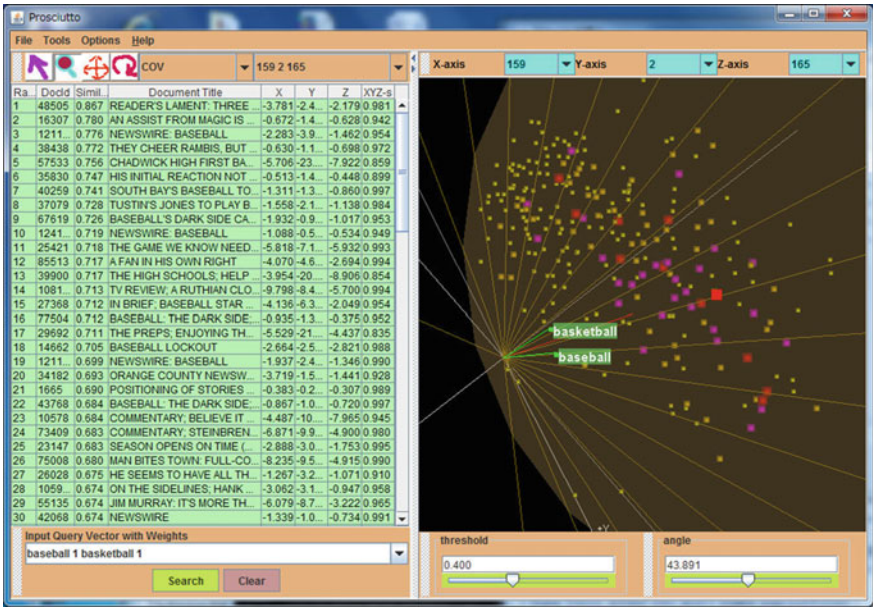

Fig. 8 Visualization of results from the query, "baseball 1 basketball 1"

directions. The clusters labeled bvy our algorithm are: {*abortion, anti-abortion, clinic, Roe*}, {*lottery, jackpot, California, ticket*}, {*AIDS, disease, virus, patient*}, {*gang, school, youth, murder*}, {*Cypress, Santiago, team, tournament*}, and {*jazz, pianist, festival, saxophonist*} for A^+, $A^-B^+B^-$, C^+, and C^-, respectively; The plus and minus sign indicate the directions in which clusters lie along a basis vector.

Additional examples that illustrate how our GUI enables quick understanding of clusters—their content (subject matter), degree of similarity between documents, and overlap structures—are given in Fig. 10a, b. Documents in the three main clusters in Fig. 10a (top) are very tightly bound together (i.e., lie very close to each other) along coordinate axes, indicating that they are very closely related. Documents in the three large clusters in Fig. 10b (bottom) are more loosely bound together than those in Fig. 10a, indicating they are only roughly similar.

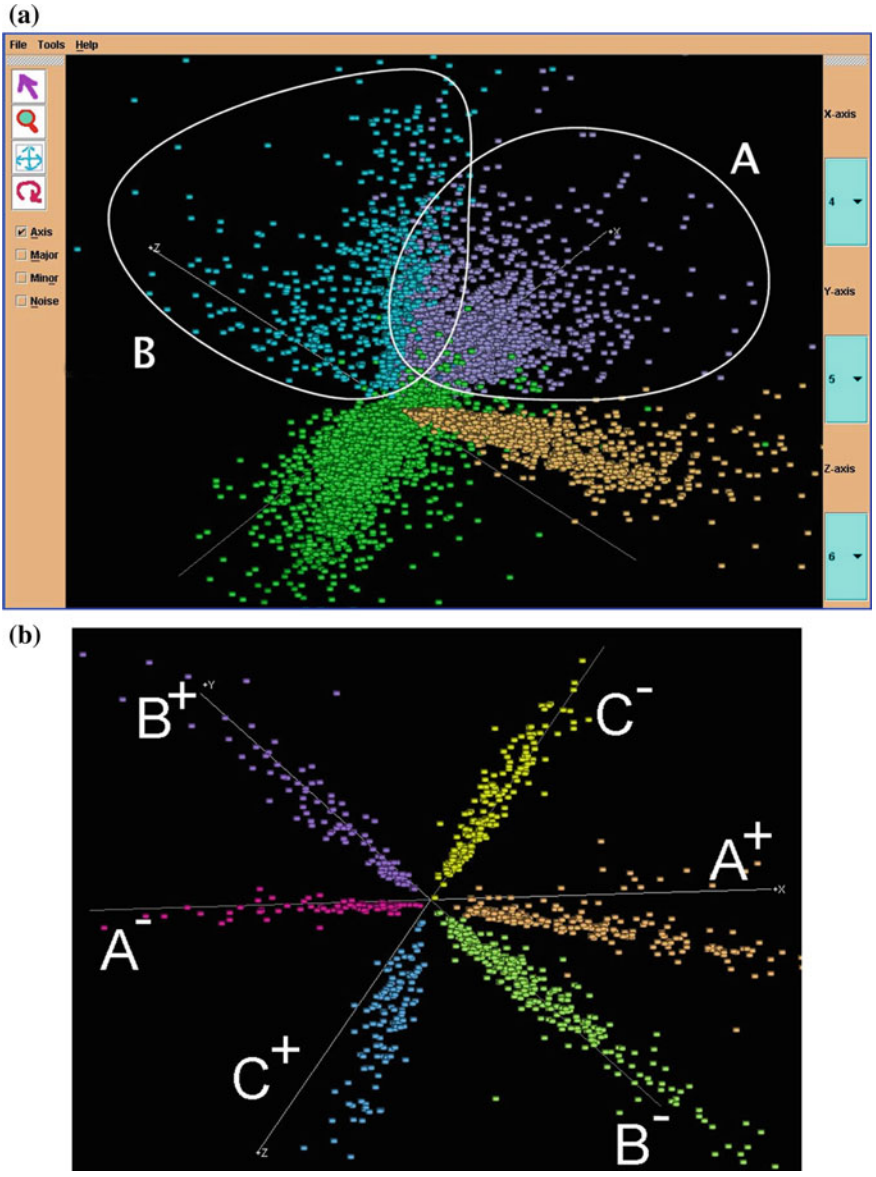

Fig. 9 **a** Visualization of overlapping clusters A and B on {school, council, tax, valley, California} with 3,433 documents and {school, art, music, film, child} with 630 documents. The clusters have 414 documents in common, with overlap ratios 12 and 75.7 %, respectively. **b** Visualization of six minor clusters on {abortion, anti-abortion, clinic, Roe}, {lottery, jackpot, California, ticket}, {AIDS, disease, virus, patient}, {gang, school, youth, murder}, {Cypress, Santiago, team, tournament}, and {jazz, pianist, festival, saxophonist} labeled A^+, A^-, B^+, B^-, C^+, and C^-, respectively

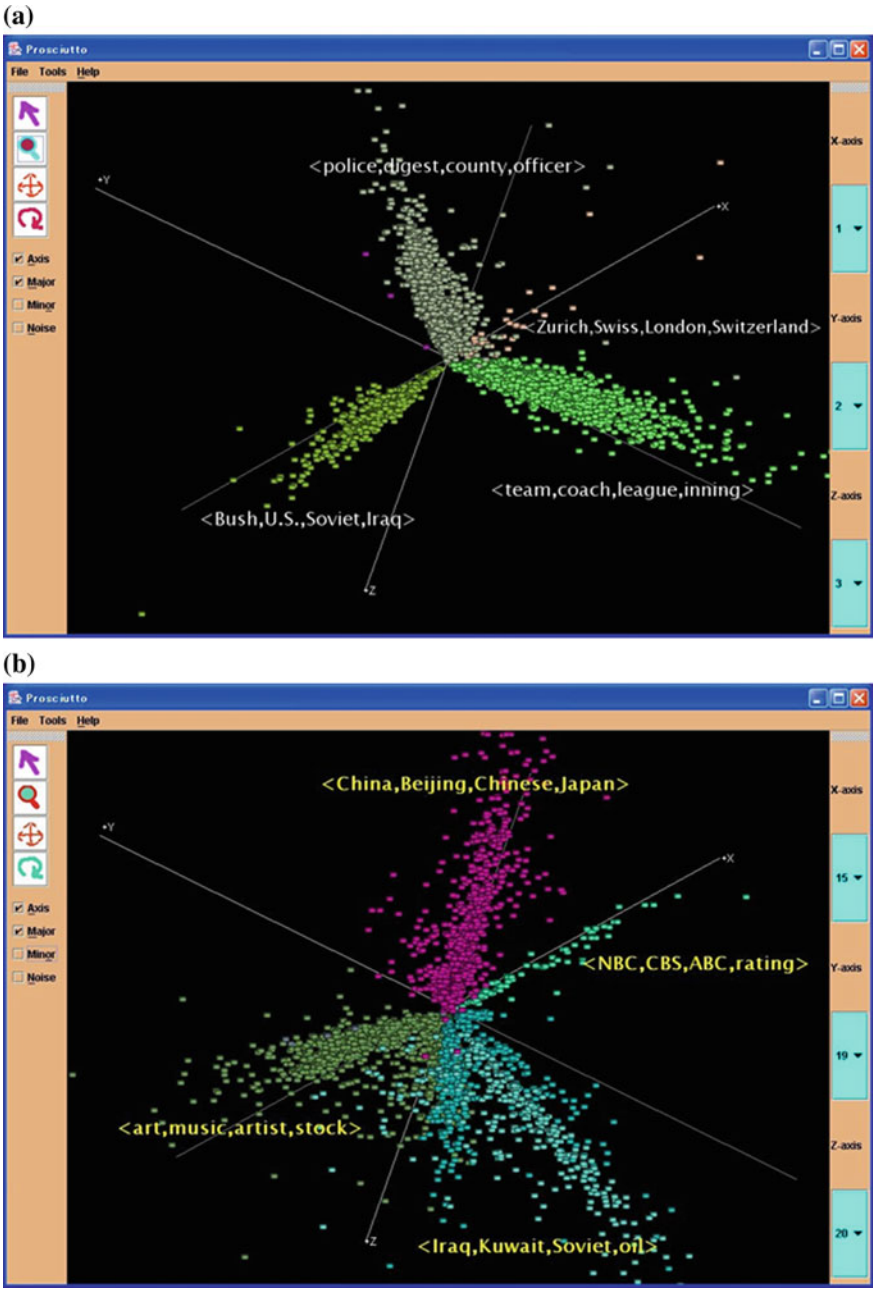

Fig. 10 **a**, **b** Visualization of overlapping clusters. Documents in the *three* main clusters in (**a**) (*top*) are very tightly bound together, while those in (**b**) (*bottom*) are more loosely bound together, indicting they are roughly similar

5 Conclusions

We proposed and implemented a new visualization system to help non-technical users understand the content of massive data sets based on their interests through a three-dimensional display of relevant documents and clusters. Our system is based on vector space modeling of documents. In the first step of the visualization process, users input keywords describing their interests (weighting is optional). Our system automatically computes recommendations for a three-dimensional subspace into which the documents will be projected and displayed, based on the proximity of three coordinate axes to the user's keyword input. More sophisticated users can specify three projection axes for the display (and skip keyword input).

We conducted implementation studies with artificially generated and real-world data that demonstrate how our interface can help non-technical users understand documents and clusters in massive databases. Specifically, the interface shows cluster sizes, distances between clusters, similarities of documents within clusters, and extent of cluster overlaps. Users who are interested in specific documents can call-up a list of relevant document titles as well as the text of the original document. Our GUI can also be used to evaluate the quality of output from different clustering algorithms or help users select a clustering algorithm for an intended purpose.

Some interesting topics for future research include development of: (1) a simple, new means to measure of quality of each cluster (quantitatively as well as quantitatively and visually); (2) a simple-to-manipulate, concise and easy-to-understand means for presenting relationship among clusters, particularly regions which have many cluster data element overlaps; (3) an extension of the system to time-series data to visualize changes in documents on user-specified topics over time, and (4) an extension of the system to automatically generate simple user-session reports with statistics on data in images that were generated and viewed.

Acknowledgments This work was conducted at IBM Research-Tokyo. The authors would like to acknowledge many helpful conversations with their colleagues. Our special thanks go out to Arquimedes Canedo, Yun Zhang, Steven Gardiner and Mike Berry for helpful suggestions on our manuscript and to Koichi Takeda and Fumio Ando for their thoughtful management and support of our work.

References

1. Achlioptas, D.: Database-friendly random projections. In: Proceeding of the ACM PODS, pp. 274–281. Santa Barbara, CA (2001)

2. Ankerst, M., Keim, D., Kriegel, H.-P.: Circle segments: a technique for visually exploring large multidimensional data sets. In: Proceeding of the IEEE Visualization, pp. 274–281. San Francisco, CA (1996)
3. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. Addison-Wesley, Reading (1999)
4. Banerjee, A., Krumpelman, C., Basu, S., Mooney, R., Ghosh, J.: Model-based overlapping clustering. In: Proceeding of the ACM KDD, pp. 532–537. Chicago, IL (2005)
5. Battle, A., Segal, E., Koller, D.: Probabilistic discovery of overlapping cellular processes and their regulation using gene expression data. In: Proceeding of the ACM RECOMB, pp. 167–176. San Diego, CA (2004)
6. Belew, R.: Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW. Cambridge University Press, Cambridge (2008)
7. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, NY (1981)
8. Blum, K., Ruhe, A.: Information retrieval using a Krylov subspace method. SIAM J. Matrix Anal. Appl. **26**, 566–582 (2005)
9. Deerwester, S., et al.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
10. Dhillon, I., Modha, D., Spangler, W.: Visualizing class structure of multidimensional data. In: Proceeding of the Symposium on Interface: Computer Science and Statistics. http://www.almaden.ibm.com/cs/people/dmodha/ (1998). Accessed 31 March 2011
11. Dunn, J.C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J. Cybern. **3**, 32–57 (1973)
12. Everitt, B., Landau, S., Leese, N.: Cluster Analysis, 4th edn. Oxford University Press, Oxford (2001)
13. Faloutsos, C., Lin, K.-I.: FastMap: a fast algorithm for indexing, data-mining and visualization of multimedia datasets. In: Proceeding of the ACM SIGMOD, pp. 163–174. San Jose, CA (1995)
14. Friedman, J., Tukey, J.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comp. **c-23**(9), 881–890 (1974)
15. Futschik, M.E., Carlisle, B.: Noise-robust soft clustering of gene expression time-course data. J. Bioinform. Comput. Biol. **3**(4), 965–988 (2005)
16. Golub, G., VanLoan, C.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
17. Hinnenburg, A., Keim, D., Wawryniuk, M.: HD-eye: visual mining of high dimensional data. IEEE Comput. Graph. Appl. **19**(5), 23–31 (1999)
18. Huang, Z., Lin, T.: A visual method of cluster validation using Fastmap. In: Terano, T., Liu, H., Chen, A. (eds.) Knowledge Discovery and Data Mining, pp. 153–164. Current Issues and New Applications, Springer, Berlin (2000)
19. Inselberg, A.: The plane with parallel coordinates. Vis. Comput. **1**(2), 69–92 (1985)
20. Jain, A., Murty, M., Flynn, P.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (2000)
21. Jolliffe, I.: Principal Component Analysis, 2nd edn. Springer, Berlin (2002)
22. Kandogan, E.: Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In: Proceeding of the ACM KDD, pp. 107–116. San Francisco, CA (2001)
23. Kobayashi, M., Aono, M.: Vector space models for search and cluster mining. In: Berry, M. (ed.) Survey of Text Mining: Clustering, Classification and Retrieval, pp. 103–122. Springer, NY (2004)
24. Kobayashi, M., Aono, M.: Exploring overlapping clusters using dynamic re-scaling and sampling. Knowl. Inf. Syst. **10**(3), 295–313 (2006)
25. Kobayashi, M., Aono, M.: Vector space models for search and cluster mining. In: Berry, M., Castellanos, M. (eds.) Survey of Text Mining, 2nd edn., pp. 103–122. Springer, Berlin (2008)
26. Kobayashi, M., Aono, M., Samukawa, H., Takeuchi, H.: Matrix computations for knowledge mining and management. J. Comput. Appl. Math. **149**, 119–129 (2002)

27. Kriegel, H.-P., Kroeger, P., Zimek, A.: Clustering high dimensional data. ACM Trans. Knowl. Discov. Data **3**(1), 1–58 (2009)
28. Krushal, J.: Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation'. In: Milton, R., Nelder, J. (eds.) Stat. Comput., pp. 427–440. Academic Press, NY (1969)
29. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceeding of the 5th Berkeley Symposium on Mathematical Statistics and Probability 1, pp. 281–297, University of California Press (1967)
30. Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press, NY (1979)
31. Nason, G.: Design and choice of projection indices. Dissertation, University of Bath, UK (1992)
32. Park, H., Jeon, M., Rosen, B.J.: Lower dimensional representation of text data based on centroids and least squares. BIT **43**(2), 1–22 (2003)
33. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philos. Mag. **2**(6), 559–572. http://stat.smmu.edu.cn/history/pearson1901.pdf (1901). Accessed 25 Nov 2011
34. Popescul, A., Ungar, L.: Automatic labeling of document clusters. http://www.cis.upenn.edu/popescul/Publications/popescul001labeling.pdf (2000). Accessed 31 March 2011
35. Rasmussen, E.: Clustering algorithms. In: Frakes, E., Baeza-Yates, R. (eds.) Information Retrieval, pp. 419–442. Prentice Hall, Englewood Cliffs (1992)
36. Rohrer, R., Silbert, J., Ebert, D.: A shape-based visual interface for text retrieval. IEEE Comput. Graph. Appl. **19**(5), 40–46 (1990)
37. Sahami, M., Hearst, M., Saund, E.: Applying the multiple cause mixture model to text categorization. In: Proceeding of the ICML, pp. 435–443. Baru, Italy (1996)
38. Salton, G. (ed.): The Smart Retrieval System. Prentice Hall, Englewood Cliffs (1971)
39. Sebrechts, M., et al.: Visualization of search results; a comparative evaluation of text, 2D, and 3D interfaces. In: Proceeding of the ACM SIGIR, pp. 3–10. Berkeley, CA (1999)
40. Segal, E., Battle, A., Koller, D.: Decomposing gene expression into cellular processes. In: Proceeding of the Pacific Symposium on Biocomputing, Lihue, HI, vol. 8, pp. 89–100. http://helix-web.stanford.edu/psb03/segal.pdf (2003). Accessed 25 Nov 2011
41. Seo, J., Shneiderman, B.: Interactively exploring hierarchical clustering results. IEEE Comput. **35**(7), 80–86 (2002)
42. Spence, R.: Information Visualization, 2nd edn. Prentice-Hall, Englewood Cliffs (2007)
43. Ware, C.: Information Visualization, 2nd edn. Morgan Kaufmann, Burlington (2004)
44. Wong, P.: Visual data mining. IEEE Comput. Graph. Appl. **19**(5), 20–21 (1999)
45. Zhao, Y., Karypis, G.: Soft clustering criterion functions for partitional document clustering. In: Proceeding of the ACM CIKM, pp. 246–247. Washington DC (2004)
46. Zhukov, L., Gleich, D.: Decomposing gene expression into cellular processes. http://www.stanford.edu/dgleich/publications/soft-clustering-pca-ica.pdf (2003). Accessed 31 March 2011